

Rational Test Virtualization Server



# Reference Guide

*Version 8.0.0*



**Note**

Before using this information and the product it supports, read the information in “Notices” on page 175.

This edition applies to version 8.0.0 of Rational Test Virtualization Server and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2001, 2012.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>About this Publication</b>	<b>vi</b>
Intended Audience	vii
Scope	vii
Typographical Conventions	vii
Contacting IBM Support	vii
<b>Introduction</b>	<b>1</b>
Functional Overview	2
Definition	2
Components	3
Supported Stub-Types	6
Virtualization Capability	6
Usage Scenarios	7
Before You Begin	9
Installing Required Software	9
Creating a Rational Integration Tester Project Results Database	9
<b>Getting Started</b>	<b>10</b>
Using Rational Integration Tester	11
Starting the Application	11
Creating a New Project	11
Navigating the User Interface	11
Using Rational Test Control Panel	12
Logging In/Out	13
Navigating the User Interface	17
<b>Creating &amp; Modifying Message-Based Stubs</b>	<b>18</b>
Introduction	19

---

Purpose.....	19
Benefits.....	19
Levels of Richness.....	19
Creating Message-Based Stubs .....	21
Empty Stub Method.....	21
Message Exchange Pattern (MEP) Method .....	24
Recording Studio Method .....	27
Modifying Message-Based Stubs .....	53
Transitions Tab.....	54
Behaviour Tab .....	76
Properties Tab .....	80
Logging Tab .....	83
Documentation Tab.....	84
<b>Creating &amp; Modifying Database Stubs .....</b>	<b>85</b>
Introduction .....	86
Purpose.....	86
Benefits.....	86
Key Concepts .....	86
Before Creating Database Stubs .....	90
Preparation and Planning.....	90
Setting Up the Schema Used for Stubbing a Physical Database .....	90
Creating Database Stubs .....	94
Recording Studio Method .....	94
Proxy Mode Change Method.....	96
Modifying Database Stubs .....	100
Editing Options Toolbar .....	101
Actions .....	102
<b>Publishing &amp; Running Stubs.....</b>	<b>108</b>
Publishing Stubs .....	109
Verifying Publication of Stubs.....	114
Running Stubs .....	116
Starting Stubs.....	116

---

---

Stopping Stubs.....	122
Modifying Running Stubs .....	124
Controlling Running Stubs.....	124
Viewing Stub Error Messages .....	126
Viewing Stub Logs .....	126
<b>Managing Agents.....</b>	<b>129</b>
Viewing All Running Agents Registered with Current Rational Test Control Panel Instance .....	130
Viewing Agents For a Specific Domain/Environment .....	132
<b>Troubleshooting.....</b>	<b>133</b>
Handling Agent Failures .....	134
Resolving Stub Log Display Problems .....	136
<b>Appendix A: Using the Data Model Editor.....</b>	<b>137</b>
Creating Data Models .....	138
Editing Data Models.....	144
Deleting Data Models .....	148
<b>Appendix B: Creating Behaviours .....</b>	<b>149</b>
Introduction .....	150
Creating a Behaviour.....	151
Defining Interfaces.....	171
Behaviour Interface .....	171
Callback Interface .....	171
Behaviour Factory .....	172
Behaviour Implementation .....	173
<b>Glossary .....</b>	<b>174</b>
<b>Notices .....</b>	<b>175</b>
Trademarks and service marks .....	178

# About this Publication

## **Contents**

### **Intended Audience**

### **Scope**

### **Typographical Conventions**

### **Contacting IBM Support**

This guide describes how to use IBM® Rational® Test Virtualization Server to create and run message-based stubs, database stubs, and rich virtualized applications.

---

## Intended Audience

This document assumes that readers are familiar with software testing, especially functional testing. Ideally, readers should also be familiar with the concepts underlying virtualized applications.

## Scope

This document describes how to use Rational Test Control Panel and Rational Integration Tester to create and run message-based stubs, database stubs, and virtualized applications.

For information about installing Rational Test Virtualization Server, refer to the following documents:

- *IBM Rational Integration Tester Agent Installation Guide*
- *IBM Rational Integration Tester Installation Guide*
- *IBM Rational Test Control Panel Installation Guide*
- *IBM Rational Integration Tester Platform Pack Installation Guide* (optional)

## Typographical Conventions

The following typographical conventions are observed throughout this document:

Type	Usage
Constant Width	Program output, listings of code examples, file names, commands, options, configuration file parameters, and literal programming elements in running text.
<i>Italic</i>	Document title names in statements that refer you to other documents. Also used to highlight concepts when first introduced.
<b>Bold</b>	Menu items in graphical user interface windows (such as Microsoft Windows-based or UNIX X Window applications) from which you select options or execute macros and functions. Submenus and options of a menu item are indicated with a “greater than” sign, such as <b>Menu &gt; Submenu</b> or <b>Menu &gt; Option</b> .

## Contacting IBM Support

To contact IBM Support, see: [www.ibm.com/contact/us/en/](http://www.ibm.com/contact/us/en/)

# Introduction

## **Contents**

### **Functional Overview**

### **Before You Begin**

This chapter provides an overview of Rational Test Virtualization Server, and outlines what you need to do before you can start using Rational Test Control Panel and Rational Integration Tester to create and run stubs.



---

## 1.1 Functional Overview

The following sections outline the purpose and the components of Rational Test Virtualization Server, and when, why, and how the application should be used.

### 1.1.1 Definition

Rational Test Virtualization Server is IBM Rational software that is used for creating, maintaining, publishing, and running message-based stubs, database stubs, and rich virtualized applications.

Stubs and virtualized applications are used to simulate services within an environment for the purposes of software development and testing. Simulating parts of an environment can often be necessary if the real services are not yet available or because they are difficult or expensive to use.

Additionally, from a testing point of view, a tester will often require simple or deterministic responses from services used by the system under test (SUT) and the only way to ensure this is to develop stubs that act in a known way.

Rational Test Virtualization Server offers extremely powerful tools to quickly create and manage stubs in large environments. However, **stubs do not simulate underlying messaging transports**. For example, although Rational Test Virtualization Server does not simulate IBM WebSphere® MQ, it can simulate a service that is accessed over WebSphere MQ and any related stubs will still be accessed by means of WebSphere MQ.

Further, in most situations, **stubs do not remove the need for an Enterprise Service Bus (ESB) or messaging software**. A common exception to this is HTTP(S) or TCP-based Web Services where a Rational Test Virtualization Server stub will act as the HTTP(S)/TCP endpoint for the message.

Rational Test Virtualization Server helps developers by enabling them to create stubs representing services that are needed for the completion of development and testing but which that may not yet exist or which are difficult and time consuming to set up.

Rational Test Virtualization Server helps testers by enabling them to stub out the dependencies of an SUT and to control the external dependencies of that SUT, so they can plan, organize, manage, and control their software testing more effectively and more efficiently.

---

### 1.1.2 Components

Rational Test Virtualization Server comprises several IBM Rational software components. The following table outlines how each of those components is used within Rational Test Virtualization Server.

---

Software component	Mandatory, optional, or conditional	Purpose
Rational Integration Tester	Mandatory	<p>This is the primary application for creating stubs. It can be regarded as the design-time environment for stubs. It can also be used for limited deployment of virtual services.</p> <p>It enables you to:</p> <ul style="list-style-type: none"><li>• Record events from the system.</li><li>• Create stubs in a variety of ways, including from recorded events, and then executing them.</li><li>• Create data models based on data in recorded messages for groups of stubs to use, facilitating the creation of richer virtual services.</li></ul> <p>For general information about using Rational Integration Tester, refer to <i>IBM Rational Integration Tester Reference Guide</i>.</p> <p>Information about using Rational Integration Tester as part of Rational Test Virtualization Server is contained elsewhere in this document.</p>
Rational Test Control Panel	Mandatory	<p>This application enables you to manage virtual services, agents, and proxies within an environment.</p> <p>Typically, after a stub has been created, it will be published from Rational Integration Tester to Rational Test Control Panel. The stub is then stored in a repository on Rational Test Control Panel. From this repository, each virtual service can be configured, deployed, and managed.</p> <p>For information about administering Rational Test Control Panel, refer to <i>IBM Rational Test Control Panel System Administration Guide</i>.</p> <p>Information about using Rational Test Control Panel as part of Rational Test Virtualization Server is contained elsewhere in this document.</p>

---

---

---

Software component	Mandatory, optional, or conditional	Purpose
Rational Integration Tester Agents	Mandatory	<p>Agents run stubs in the Rational Test Virtualization Server environment and are deployed on one or more computers within the environment</p> <p>Agents act as hosts for virtual services, enabling them to be deployed to different locations across a network.</p> <p>They are registered with, receive instructions from, and report log data to, Rational Test Control Panel.</p> <p>For general information about using Rational Integration Tester Agents, refer to <i>IBM Rational Integration Tester Reference Guide</i> and <i>IBM Rational Performance Test Server Reference Guide</i>.</p> <p>Information about using agents with Rational Test Virtualization Server is contained elsewhere in this document.</p>

---

Software component	Mandatory, optional, or conditional	Purpose
Rational Integration Tester Platform Pack (Proxies)	Conditional, depending on your testing requirements	<p>This component is required if you want to record:</p> <ul style="list-style-type: none"> <li>• Live HTTP(S)- or TCP-based messages and route those messages automatically to either the live system or to a stub without having to constantly change the configuration of any client or server applications.</li> <li>• SQL events and/or stub databases that use JDBC connections.</li> </ul> <p>Rational Integration Tester proxies are registered with, receive instructions from, and report log data to, Rational Test Control Panel.</p> <p>For general information about setting up a Rational Integration Tester HTTP proxy for use, refer to <i>IBM Rational Integration Tester Reference Guide for HTTP &amp; Web Services</i>. For general information about setting up a Rational Integration Tester HTTP proxy for TCP traffic, refer to <i>IBM Rational Integration Tester Reference Guide for TCP/UDP Sockets</i>.</p> <p>For information about using the Rational Integration Tester proxies to record HTTP(S)/TCP traffic, refer to <i>IBM Rational Integration Tester Reference Guide</i>.</p> <p>Information about using the Rational Integration Tester proxies to route HTTP(S)- and TCP-based traffic to stubs automatically is contained elsewhere in this document.</p> <p>Rational Integration Tester JDBC proxies are registered with, receive instructions from, and report log data to, Rational Test Control Panel.</p> <p>For information about using the Rational Integration Tester JDBC proxy to record SQL events, refer to <i>IBM Rational Integration Tester Reference Guide</i>. Information about using the Rational Integration Tester JDBC proxy to stub databases is contained elsewhere in this document.</p>

**NOTE:** Depending on whether Rational Test Virtualization Server users are recording, or stubbing (virtualizing), or both, the communications among all these components will be slightly different.

---

### 1.1.3 Supported Stub-Types

Stubs and virtualized applications can be simple or rich. Rational Test Virtualization Server provides tools to create stubs and virtualized applications to the level required.

The following table summarizes the various types of stubs that can be created by Rational Test Virtualization Server.

Stub-Type	Description
Basic	There is a hard-coded single response for each specific input.
Non-deterministic	There are “n” hard-coded responses. A message switch is used to “switch” the response based on the input message.
Data-driven (parameterized)	There is input and/or output data specified in external data sources, for example, databases or spreadsheet files.
Data model-driven	There is input and/or output data in a data model that includes relationships among those data items.
Behaviour-driven	Stubs can be extended with “behaviours” that are Java plug-ins that can respond to messages and proactively cause the stub to behave in a particular way.  IBM supplies some behaviours with Rational Integration Tester but customers can write their own. For example, a behaviour can be used to make a stub act as a market data feed source.
Deterministic	This comprises a series of Receive Request/Send Response or Subscribe/Publish actions that are based on the message used to create the stub.

### 1.1.4 Virtualization Capability

Rational Test Virtualization Server’s virtualization functionality is more granular than that of a Virtual Machine (VM) because Rational Test Virtualization Server can virtualize an application or database **or** just part of an application or database.

In contrast, Virtual Machines (VMs):

- Are designed to virtualize an entire machine.
- Require licenses for their applications and are usually not maintained by test teams.
- Are less flexible for testing purposes whereas a virtualized application in Rational Test Virtualization Server can be manipulated easily to fit your testing purposes. For example, Rational Test Virtualization Server enables you to created virtualized

---

applications that can send erroneous data for negative testing of a system under test.

### 1.1.5 Usage Scenarios

During software development projects, functional and non-functional requirements can change quickly, and test environments and applications are often in high demand from other teams.

Rational Test Virtualization Server can help because it enables you to:

- Continue testing when test environments are unavailable or not yet built.
- Test earlier and more often, reducing the cost of defects.
- Control the responses from services, enabling you to force the behaviour of the SUT.

In addition, Rational Test Virtualization Server is designed to be used in all software test phases from unit testing to user acceptance testing:

- During each test phase, you should aim to test in as complete a way as possible. For example, when unit testing individual operations or services, you may not always have the interfacing components available to test against. Rational Test Virtualization Server can be used to virtualize those interfacing components.
- As you proceed from unit testing to integration testing and onwards, you will probably want to introduce more “real” components into the system under test. Virtualization enables you to reduce any risks that may be associated with the introduction of those real components.

The following table outlines example scenarios where Rational Test Virtualization Server could be used.

Scenario	Description
Your testing project may be heavily reliant on integration with third parties <b>or</b> you want to control all systems with which the system under test (SUT) will communicate	Integration with third parties can be immensely frustrating and costly.  Rational Test Virtualization Server can virtualize third party interfaces to enable you to test on your own terms according to your schedule.

---

---

Scenario	Description
You have integration testing dependencies	<p>Parallel development can sometimes mean that some projects may not be ready to begin integration testing when your project is ready.</p> <p>Rational Test Virtualization Server enables you to virtualize interfaces (even before they have been built) and continue testing.</p>
You want to run training or demonstration instances of applications without access to a large infrastructure	<p>For training or demonstration purposes, you may not require access to a production-size version of the system under test. In addition, you may not require access to any “downstream” applications.</p> <p>Rational Test Virtualization Server can virtualize and simplify interfaces, ensuring that training or demonstration exercises do not impact any production systems.</p>
You want to test a database-dependent application with scrubbed and isolated data	<p>Rational Test Virtualization Server can simulate databases as well applications. This means that you will have full control of all data to be used during testing.</p>
You want to provide a test system where none currently exists	<p>It may be too expensive to build a test environment and it may take too long to build it. Alternatively, there may be a test environment but it is being used by another team for the duration of your project.</p> <p>Rational Test Virtualization Server substitutes for the “absent” test environment by virtualizing applications.</p>

---

---

## 1.2 Before You Begin

Before you can use Rational Test Virtualization Server, you must:

1. Install the Rational Test Control Panel and Rational Integration Tester applications and at least one Rational Integration Tester Agent.
2. Create a Rational Integration Tester project results database (optional depending on requirements).

The following sections outline these tasks.

### 1.2.1 Installing Required Software

Although certain stubbing-related tasks can be accomplished within Rational Integration Tester, Rational Test Control Panel is required for managing virtual services, agents, and proxies within a large environment.

For information about installing Rational Test Control Panel, refer to *IBM Rational Test Control Panel Installation Guide*. For information about installing Rational Integration Tester, refer to *IBM Rational Integration Tester Installation Guide*.

**NOTE:** Rational Test Control Panel and Rational Integration Tester do **not** have to be installed on the same computer.

### 1.2.2 Creating a Rational Integration Tester Project Results Database

All Rational Integration Tester test data is stored in a project results database. If you want to view the output of any stubs that you will create and run, you must create a project results database after Rational Integration Tester is installed. However, if you want only to create and run stubs, creating a project results database is optional.

For information about creating and configuring a project results database, refer to *IBM Rational Integration Tester Installation Guide*.



# Getting Started

## **Contents**

**Using Rational Integration Tester**

**Using Rational Test Control Panel**

This chapter describes how to start using Rational Integration Tester and Rational Test Control Panel, which are the two primary components of Rational Test Virtualization Server.

---

## 2.1 Using Rational Integration Tester

**NOTE:** If you have previous experience of using Rational Performance Test Server and/or Rational Integration Tester, you may skip this section and proceed to [Using Rational Test Control Panel](#).

Rational Integration Tester is used for the following activities:

- Creating a model of the system under test.
- Recording events from the system.
- Creating, designing, and testing stubs before publishing them to Rational Test Control Panel.

The following sections outline how to start using the Rational Integration Tester application.

### 2.1.1 Starting the Application

For detailed instructions about how to start Rational Integration Tester, refer to *IBM Rational Integration Tester Getting Started Guide* or *IBM Rational Integration Tester Reference Guide*.

### 2.1.2 Creating a New Project

For detailed instructions about creating projects in Rational Integration Tester, refer to *IBM Rational Integration Tester Getting Started Guide* or *IBM Rational Integration Tester Reference Guide*. You must ensure that the correct Rational Test Control Panel URL is specified on the second screen of the Create New Project wizard.

### 2.1.3 Navigating the User Interface

Rational Integration Tester's user interface is “dockable”, so sub-windows are attached or “docked” to one side of the application's workspace. The workspace comprises six perspectives, which can be selected from the **Perspectives** toolbar.

For more information about Rational Integration Tester's user interface, refer to *IBM Rational Integration Tester Getting Started Guide* or *IBM Rational Integration Tester Reference Guide*.

---

## 2.2 Using Rational Test Control Panel

**NOTE:** This section is a brief overview of Rational Test Control Panel and it is intended primarily for Rational Test Virtualization Server users who are not also Rational Test Control Panel administrators. The use of Rational Test Control Panel is also discussed in [Publishing & Running Stubs](#). For additional information about using Rational Test Control Panel, Rational Test Control Panel administrators should refer to *IBM Rational Test Control Panel System Administration Guide*.

Rational Test Control Panel is used for managing virtual services, agents, and proxies within an environment.

After a stub has been created, it can be published from Rational Integration Tester to Rational Test Control Panel. Therefore, you may not need to use Rational Test Control Panel until you have created at least one stub.

All Rational Test Control Panel users can accomplish the following tasks with Rational Test Control Panel:

- Change their login passwords (depending on how Rational Test Control Panel security is configured).
- View the Rational Test Virtualization Server “landscape” and start and stop stubs.
- View and modify scheduled tests.

Rational Test Control Panel administrators can accomplish additional tasks, such as managing users and domains, removing published stubs, and viewing Rational Test Control Panel logs.

The following sections outline how to start using the Rational Test Control Panel application.

---

## 2.2.1 Logging In/Out

The following sections describe how to log into, and log out from, Rational Test Control Panel; and how to change your Rational Test Control Panel login password.

### 2.2.1.1 Logging In

Depending on the security model that Rational Test Control Panel is using, you may have to **request** a Rational Test Control Panel **address** (host name or IP address, and port number), **user name**, and **password** from a Rational Test Control Panel administrator before you can log into the application.

In addition, there may be more than one Rational Test Control Panel instance in your test environment, so you may need to seek access to more than one Rational Test Control Panel instance.

To log into Rational Test Control Panel:

1. Open a web browser and enter a URL of the following format:

```
http://<Host Name or IP Address Provided by Rational Test  
Control Panel Administrator>:<Port Number Provided by Rational  
Test Control Panel Administrator>/GHServer/
```

Alternatively, if you are using Rational Integration Tester, clicking **Open** next to the **URL** field on the **Server Settings** tab of the Project Settings dialog box (which is opened by clicking **Project > Project Settings** on the menu bar) opens your computer's default web browser with Rational Test Control Panel's default URL displayed in the browser's address bar (you may have to edit the default URL before clicking **Open**).

---

If Rational Test Control Panel's built-in security functionality is being used, Rational Test Control Panel's Login screen is displayed.



Otherwise, Rational Test Control Panel's application window is displayed and there is no need to enter any login details.

2. In the **Username** and **Password** fields, enter your user name and password (provided by a Rational Test Control Panel administrator).
3. Click **Log in**.

**NOTE:** If Rational Test Control Panel's built-in security functionality is being used, you will be prompted to change your password after you have logged into Rational Test Control Panel for the first time. (For information about this, refer to [Changing Your Login Password](#).)

---

Rational Test Control Panel's application window is displayed.



**NOTE:** If you are a Rational Test Virtualization Server user, the **VIE** (Virtual Integration Environment) icon and **VIE** navigation link should be displayed on the application window. If they are not displayed, contact a Rational Test Control Panel administrator.

---

### 2.2.1.2 Logging Out

To log out from Rational Test Control Panel:

1. Click **LOGOUT** on the upper right corner of Rational Test Control Panel's application window.

A confirmation prompt is displayed.



2. Click **OK**.

You have now quitted the application.

**NOTE:** If Rational Test Control Panel's security functionality is disabled, the **LOGOUT** button is not displayed, so quitting your web browser will enable you to quit the Rational Test Control Panel application.

### 2.2.1.3 Changing Your Login Password

If Rational Test Control Panel's built-in security functionality is being used, you can change your Rational Test Control Panel login password at any time irrespective of your Rational Test Control Panel user privileges.

To change your login password:

1. In the **Password** and **Re-enter password** fields on the **Change Password** tab on the **Administration** page, enter and re-enter a new password (your user name should be displayed on the tab).

**NOTE:** A password must be unique but it can contain spaces and there is no limit on the number of characters that can be used.

2. Click **Change Password**.

**NOTE:** The **Change Password** button becomes unavailable if the passwords entered in the **Password** and **Re-enter password** fields are different.

Your password is changed. A status message is displayed to confirm this.

---

## 2.2.2 Navigating the User Interface

After logging into Rational Test Control Panel successfully, Rational Test Control Panel's **Home** page is displayed (for information about this, refer to [Using Rational Test Control Panel](#)).

The following table describes how to use the screen controls on the **Home** page.

Clicking the...	Displays...	For more information, refer to...
Logo or <b>Home</b> navigation link	The <b>Home</b> page.	(Not applicable)
<b>Scheduling</b> icon or navigation link	The <b>Scheduling</b> page.	<i>IBM Rational Integration Tester Reference Guide</i>
<b>Agents</b> icon or navigation link	The <b>Agents</b> page.	<i>IBM Rational Test Control Panel System Administration Guide</i>
<b>VIE</b> icon or navigation link	The <b>VIE</b> page.	<a href="#">Publishing &amp; Running Stubs</a>
<b>Administration</b> icon or navigation link	The <b>Administration</b> page.	<a href="#">Changing Your Login Password</a> <b>NOTE:</b> The <b>Administration</b> page provides access to Rational Test Control Panel to various Rational Test Control Panel features, including activity and audit logs. For information about these logs, Rational Test Control Panel administrators should refer to <i>IBM Rational Test Control Panel System Administration Guide</i> .
<b>LOGOUT</b> button <b>NOTE:</b> This button is not displayed if Rational Test Control Panel's security functionality is disabled.	A logout confirmation prompt.	<a href="#">Logging Out</a>
<b>ABOUT</b> button	Software version information and IBM contact information.	(Not applicable)

---



# Creating & Modifying Message-Based Stubs

## **Contents**

### **Introduction**

### **Creating Message-Based Stubs**

### **Modifying Message-Based Stubs**

This chapter describes how to create and maintain message-based stubs.

---

## 3.1 Introduction

The following sections provide an introduction to message-based stubs.

### 3.1.1 Purpose

A message-based stub is essentially a Rational Integration Tester resource that listens for incoming messages on a particular transport. For example, a stub may be subscribed to an IBM WebSphere MQ message queue waiting for messages to arrive or it could be a Web Service waiting for a SOAP message to arrive over an HTTP connection.

A stub can reply to specific messages or message-types, matched according to incoming filters, by using Message Case actions (for information about those actions, refer to *IBM Rational Integration Tester Reference Guide*).

### 3.1.2 Benefits

Message-based stubs enable you to simulate services that you are unable to use or that may otherwise be unavailable.

For example, if you are testing a TIBCO BusinessWorks process but you do not always have access to TIBCO Designer, you can record the process events, create a stub from them, and then run the stub in place of TIBCO Designer.

Stubs can use most Rational Integration Tester transports and schemas, including CHIPS, COBOL Copybook, Fedwire, FIX, SWIFT, and XML. (Some Rational Integration Tester transports cannot be used in stubs.)

### 3.1.3 Levels of Richness

A message-based stub can be simple or rich. For example, you might want to simulate a single service that utilizes a single message case and the default case. Alternatively, you might want to simulate a Web Service or some other point-to-point operation that includes multiple message-types or operations.

The basic principles for simple and rich message-based stubs are the same because a specific reply is sent when a specific message-type is received, but there are some important differences:

- A simple message-based stub will receive a message and (optionally) validate its contents. Based on the validation results or the fact that the message was received, the stub can return some static response, for example, a simple log action, a reply message, and so on.

- 
- In contrast, a rich message-based stub can receive incoming messages and, based on the specific contents of those messages, execute out a more extensive set of actions. For example, it might look up data in a database or spreadsheet, update one or more records in a database, utilize failure paths, and so on. Through the use of message cases and other actions, a stub can be configured to generate responses in an intelligent manner.

---

## 3.2 Creating Message-Based Stubs

In Rational Test Virtualization Server, message-based stubs can be created by using any of the following methods:

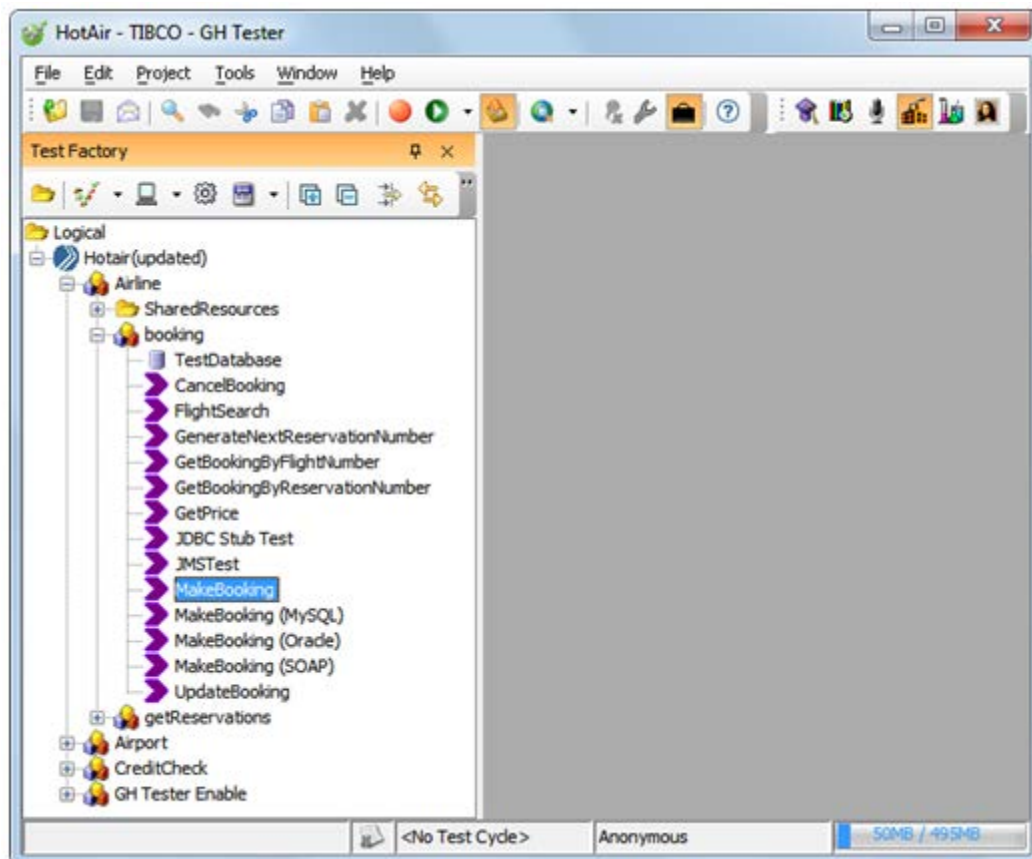
- Create an empty stub and configure it manually.
- Create a stub using Rational Integration Tester's Message Exchange Pattern (MEP) wizard.
- Create stubs from recorded events by using the Recorded Events wizard.

The following sections describe how to use each of these methods.

### 3.2.1 Empty Stub Method

To create a new empty stub:

1. Open Rational Integration Tester's Test Factory perspective.
2. Select the operation for which you want to create the stub.



- 
3. Click the **Create New Stubs** button (🖨️) on the Test Factory window's toolbar.

Alternatively, right-click the operation and click **New > Stubs > Stub** on the shortcut menu.

**NOTE:** If at least one stub has already been created, you can right-click the **Stubs** folder or an existing stub in that folder and select **New > Stub**.

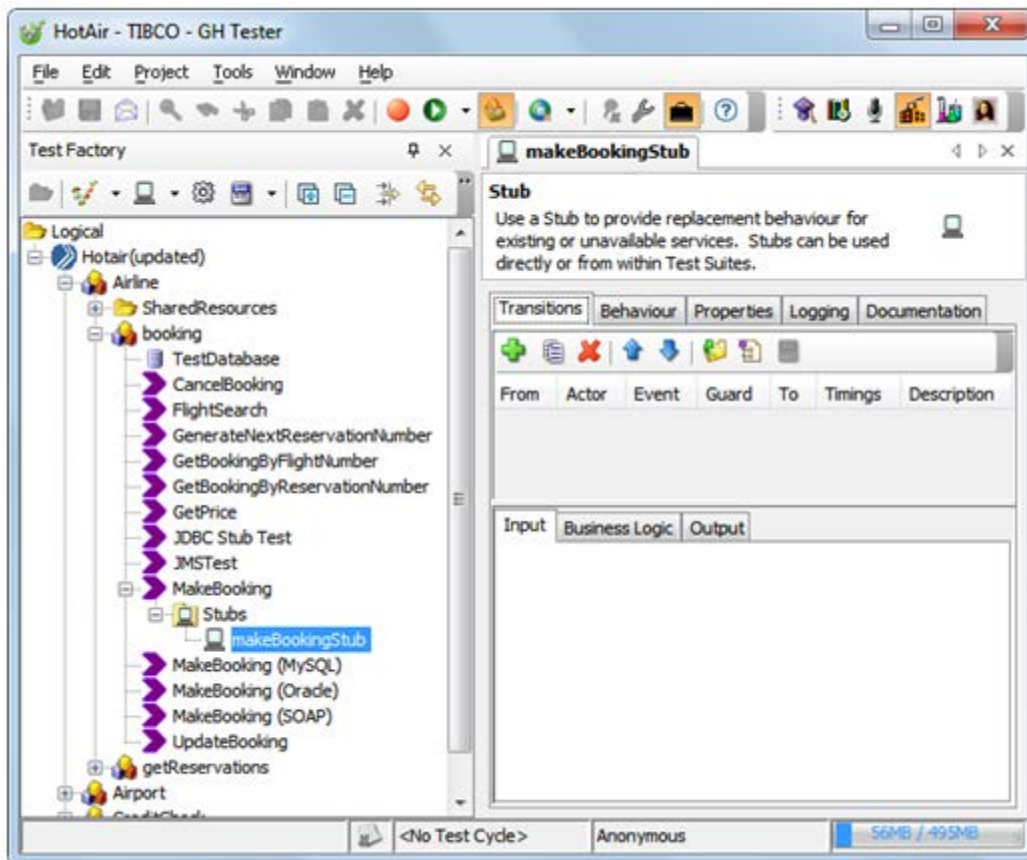
The Create new Stub dialog box is displayed.

4. In the **Name** field, enter a name for the stub.



5. Click **OK**.

The stub is opened for editing.



For information about using the Stub Editor to modify or enhance an empty stub, refer to [Modifying Message-Based Stubs](#).

6. After making any desired changes to your stub, you can save it by clicking the **Save** button (💾) on Rational Integration Tester's toolbar.

Alternatively, click **File > Save** on the menu bar or press CTRL+S.

The stub is now ready to run.

For information about running message-based stubs, refer to [Publishing & Running Stubs](#).

---

### 3.2.2 Message Exchange Pattern (MEP) Method

To create a stub by using a selected operation's MEP properties:

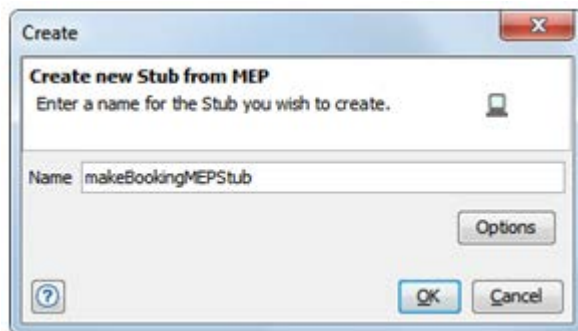
1. Open Rational Integration Tester's Test Factory perspective.
2. Click the **arrow** button (↵) next to the **Create New Stubs** button (🔧) on the Test Factory window's toolbar.

Alternatively, right-click the operation where you want to create the stub and select **New > Stubs > Stub using MEP** on the shortcut menu.

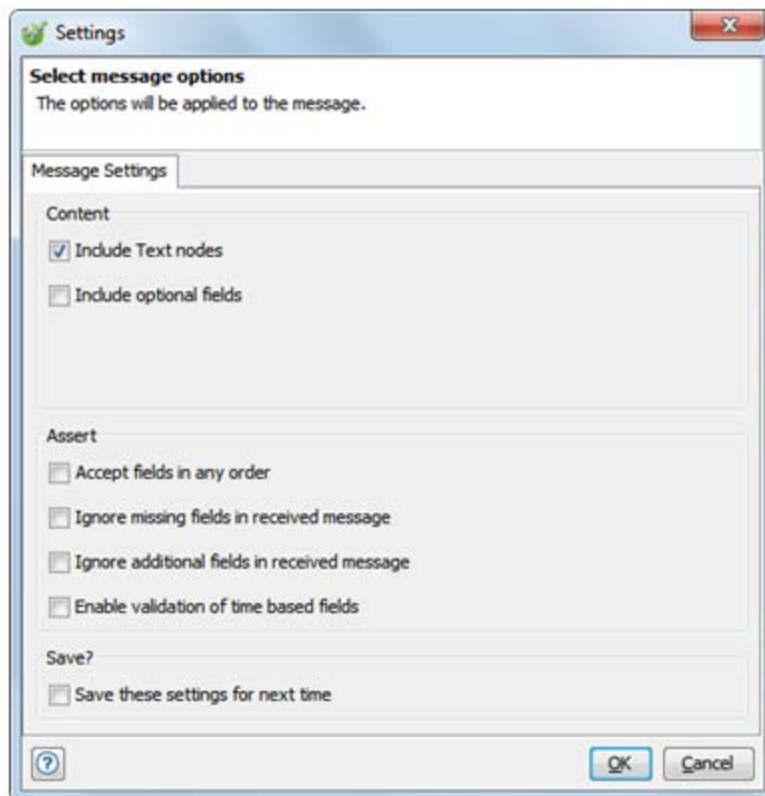
**NOTE:** If at least one stub has already been created, you can right-click the **Stubs** folder or one of the existing stubs and select **New > Stub using MEP**.

The Create new Stub from MEP dialog box is displayed.

3. In the **Name** field, enter a name for the stub.



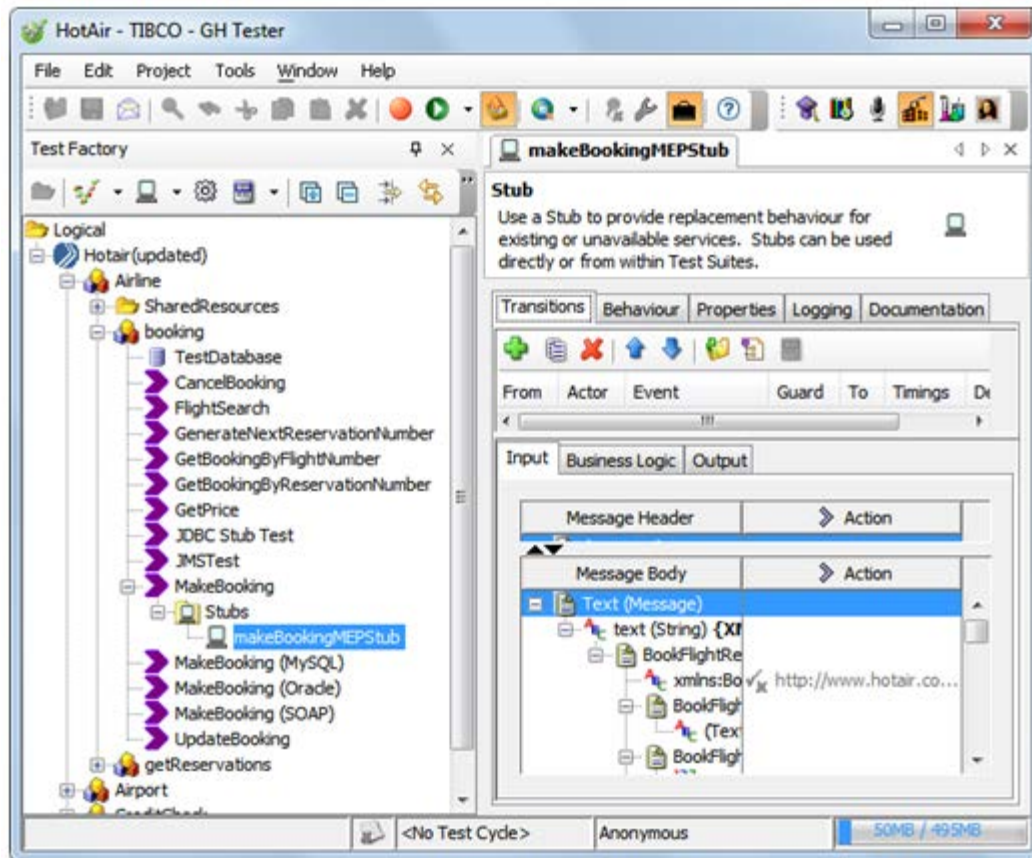
4. **Optional:** Click **Options** to open the Select message options dialog box, which enables you to enter additional settings.



5. Click **OK**.

The stub is opened for editing.





For information about using the Stub Editor to modify or enhance an “MEP stub”, refer to [Modifying Message-Based Stubs](#).

6. After making any desired changes to your stub, you can save it by clicking the **Save** button (💾) on Rational Integration Tester’s toolbar.

Alternatively, click **File > Save** on the menu bar or press CTRL+S.

The stub is now ready to run.

For information about running message-based stubs, refer to [Publishing & Running Stubs](#).

---

### 3.2.3 Recording Studio Method

This is the most effective method of creating a message-based stub if you have access to “live” systems from which you can record events, and it enables you to create very quickly stubs that behave like real services.

Rational Integration Tester’s Recording Studio perspective provides Event Monitors, which enable you to determine which parts of the environment you wish to record. For example, you might want to record events relating to specific parts of the system’s infrastructure, or you might want to record events relating to specific services that use the system’s infrastructure. After you have decided what you want to record, you can start recording events.

**NOTE:** For general information about using Rational Integration Tester’s Recording Studio, including the use of Rational Integration Tester proxies for recording events, refer to *IBM Rational Integration Tester Reference Guide*.

The environment must be modelled in Rational Integration Tester’s Architecture School perspective (Logical and Physical Views) before it can be recorded. However, for many technologies, you need to model only the transports (for example, a web server or an IBM WebSphere MQ Queue Manager), so there is no need to define any operations.

While events are being recorded, they are displayed on the Recording Studio perspective’s Events View window; and you can add or remove Event Monitors, and filter which events are displayed on the Events View window by selecting different monitors on the Event Monitors window.

**NOTE:** Recording events does not usually interfere with the operation of the system under test. For many (but not all) transports, events will still be dealt with in the same way that they would have been if Rational Integration Tester was not being used. The only difference is that the events will be accessible through Rational Integration Tester.

---

After you have completed recording events, you can use them to create any of the resource-types outlined in the following table by saving the events and using the Save wizard provided.

Resource-Type	Description
Unit tests	The data within recorded events may be hard-coded into a unit test.
Integration tests	The data within recorded events may be hard-coded into an integration test.
Stubs	The data within recorded events may be hard-coded into a basic stub that always returns the same response.
Triggers	<p>Events may be reused in the form of triggers, which enable you to stimulate the system under test directly from Rational Integration Tester.</p> <p>You can then record what happens in response but you must bear in mind that this will not necessarily be the same as what happened when you created the triggers, and you will not validating any events recorded in response to the triggers.</p> <p>Thus, you can send events to the system under test and bypass the GUI layer (and any other layers of the system that you might want to bypass) and determine how the system reacts to various inputs.</p>
Requirements	<p>You may want to save a message for subsequent use but not specify how it will be used.</p> <p>You could choose to save the message as an example message that can be viewed in Rational Integration Tester's Requirements Library and (optionally) imported into other Rational Integration Tester resources.</p>
Operations	If you have an incomplete model of the system under test, events recorded from the transports within the system can enable you to create new operations within your system model.
Test data sets	The data within recorded events may be entered into a data set, such as a comma-separated value (CSV) file, or a data model, which maps the relationships among data items in the system under test.

---

The following sections describe how to use the Recording Studio perspective to create the following message-based stub-types:

- Basic (hard-coded)
- Data-driven (parameterized)
- Data model-driven
- Deterministic

**NOTE:** For more information about using the Recording Studio perspective to create tests, triggers, requirements, and operations, refer to *IBM Rational Integration Tester Reference Guide*. For general information about stub-types, refer to [Supported Stub-Types](#).

---

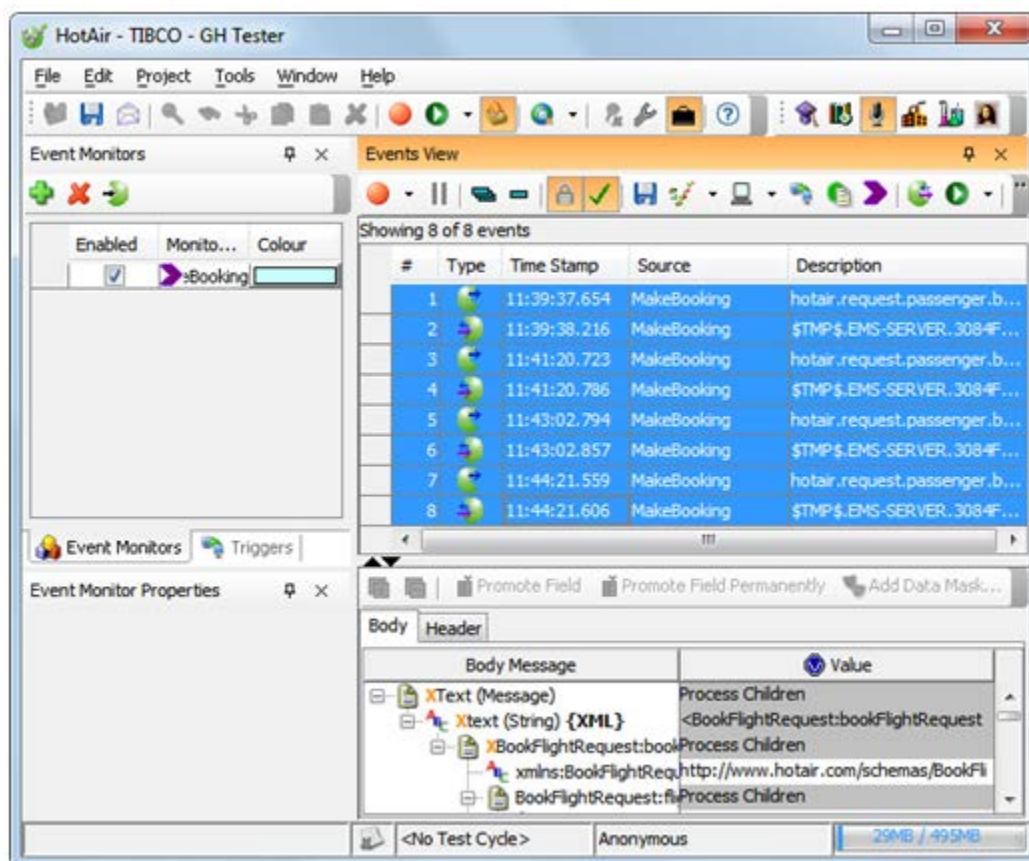
### 3.2.3.1 Creating Basic Stubs

A basic stub is a stub that always returns the same response. It does not execute any other operations, such as performing calculations, looking up any data, or making any decisions.

To create a basic (hard-coded) stub from recorded events:

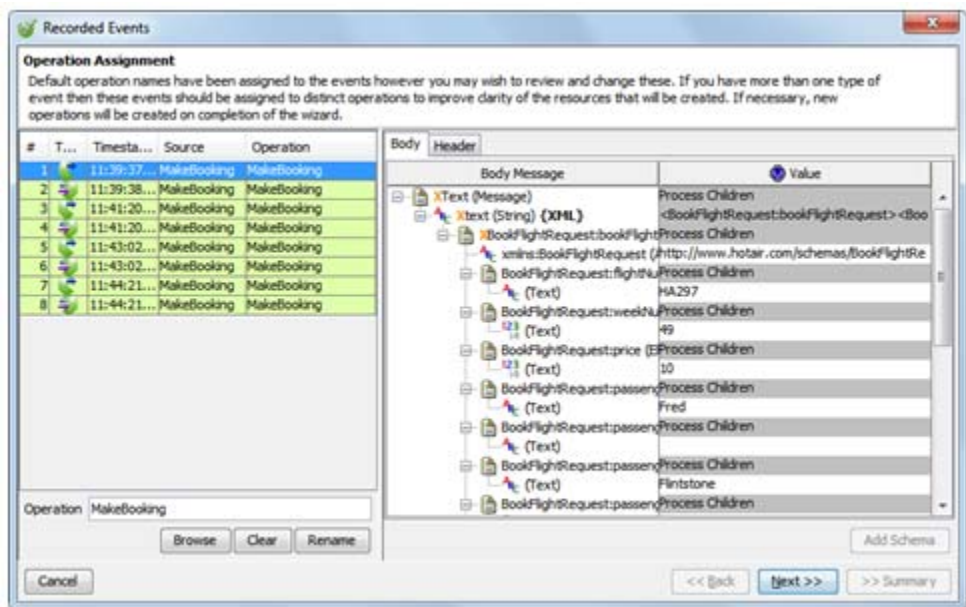
1. In Rational Integration Tester's Recording Studio perspective, select one or more recorded events on the Events View window.

**NOTE:** You should select events that represent the message exchanges that you wish to simulate. You can choose multiple message exchanges because Rational Integration Tester will distinguish among them and create appropriate operations. Thus, a stub created in Rational Integration Tester can simulate more than one operation.



**NOTE:** For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

- Click the **Save Stub from selected events** button (📄) on the Events View toolbar.  
Alternatively, right-click the messages and click **Save Stub** on the shortcut menu.  
The Operation Assignment screen of the Recorded Events wizard is displayed.



The Operation Assignment screen, which is the third screen of the wizard, enables you to modify (if you wish) the operation and the events associated with the stub.

**NOTE:** The first and second screens of the Recorded Events wizard are the Resource Type and Data Storage screens. The screens are displayed only if you click the **Save** button (💾) or press CTRL+S. Clicking the **Save Stub from selected events** button (📄) bypasses the Resource Type and Data Storage screens.

The following table describes how to use the Operation Assignment screen.

To...	Do this...
Select a different operation in the current Rational Integration Tester project	<ol style="list-style-type: none"> <li>Click <b>Browse</b> to open the Select a Resource dialog box.</li> <li>Select a different operation.</li> <li>Click <b>OK</b> to close Select a Resource dialog box.</li> </ol>

---

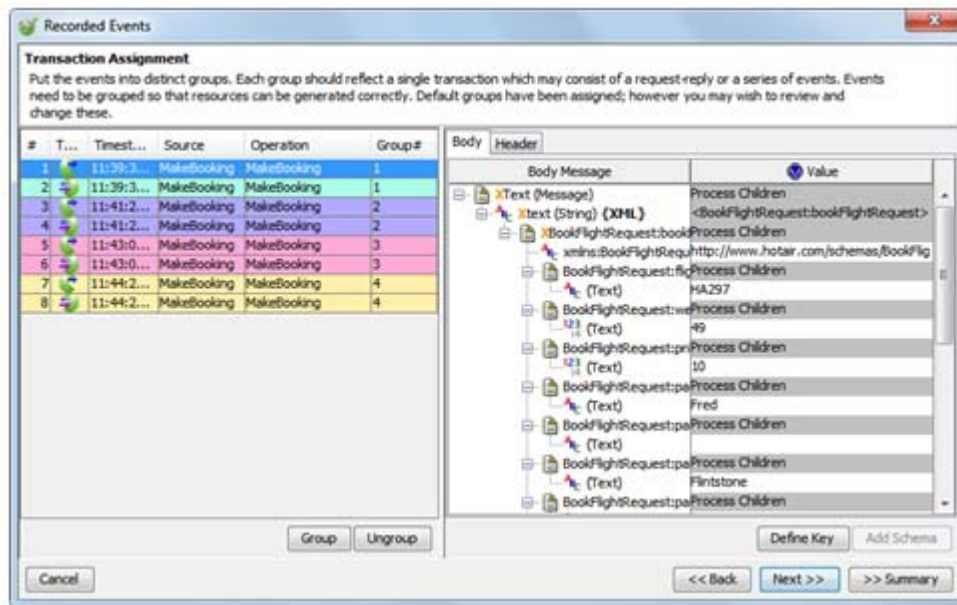
---

To...	Do this...
Select a different operation that is not in the current Rational Integration Tester project	<ol style="list-style-type: none"><li>1. Click <b>Clear</b>. Alternatively, type over text in the <b>Operation</b> field.</li><li>2. In the <b>Operation</b> field, enter the name of a new operation. (The operation will be created after you have completed using the Recorded Events wizard.)</li><li>3. Click <b>Rename</b>.</li><li>4. If you have not selected all the events displayed on the Operation Assignment screen for assignment to the new operation, you are prompted to confirm whether you want any non-selected events on the screen to be assigned to the new operation.</li></ol>
Apply a different schema to a recorded event	<ol style="list-style-type: none"><li>1. On the left side of the screen, select the event that you want to modify.</li><li>2. On the right side of the screen, click the <b>Body</b> and/or <b>Header</b> tab (as appropriate).</li><li>3. Select a field.</li><li>4. Click <b>Add Schema</b> to display the Select Schema dialog box. (For information about selecting message schemas, refer to <i>IBM Rational Integration Tester Reference Guide</i>.)</li><li>5. Clicking <b>Finish</b> on the Select Schema dialog box prompts you to confirm that you want to apply the selected schema to the selected event.</li></ol>
Close the wizard without making any changes	Click <b>Cancel</b> .
Move to the (next) Transaction Assignment screen of the wizard	Click <b>Next</b> .

---

**NOTE:** If you are creating a stub for multiple operations, Rational Integration Tester will attempt to verify that all selected recorded events are associated with the correct operations.

The Transaction Assignment screen, which is the fourth screen of the Recorded Events, enables you to group events into transactions. A transaction may consist of request-reply or a series of events.



The following table describes how to use the Transaction Assignment screen.

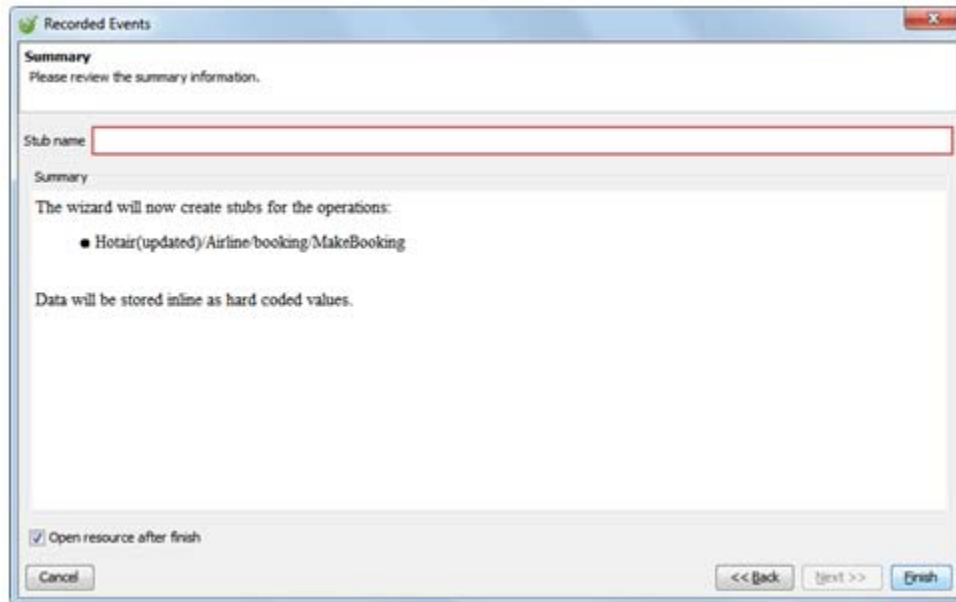
To...	Do this...
Group two or more recorded events into a single transaction	<ol style="list-style-type: none"> <li>1. On the left side of the screen, select the events that you want to group into a transaction.</li> <li>2. Click <b>Group</b>. The <b>Group#</b> field of the selected events changes.</li> </ol>
"Ungroup" a transaction	<ol style="list-style-type: none"> <li>1. On the left side of the screen, select the events that you want to ungroup.</li> <li>2. Click <b>Ungroup</b>. The <b>Group#</b> field of the selected events changes.</li> </ol>



To...	Do this...
<p>Add one or more key fields to one or more selected events displayed on the screen</p> <p>Defining a key field enables you any events on the screen that contain that field but that grouping does not supersede or overwrite any groups of transactions created by selecting events and clicking <b>Group</b></p>	<ol style="list-style-type: none"> <li>1. On the left side of the screen, select the event that you want to modify.</li> <li>2. On the right side of the screen, click the <b>Body</b> and/or <b>Header</b> tab (as appropriate).</li> <li>3. Select a field.</li> <li>4. Click <b>Define Key</b>. If the field's value is also in any other fields of other events displayed on the screen, the Define Key dialog box is displayed. Otherwise, an error message is displayed.</li> <li>5. On the Define Key dialog box, clear the check boxes of any other events that do not have the same logical value as the field you selected.</li> <li>6. <b>Optional:</b> In the <b>Save as a Field Type called</b> field, enter a comment for key field. Any comment you enter will be displayed in the Architecture School perspective's Rule Cache.</li> <li>7. Click <b>OK</b> to save your changes and to close the Define Key dialog box.</li> </ol> <p>Any messages on the screen that are not already grouped into transactions and that contain the key field that you have defined are grouped.</p>
<p>Apply a different schema to a recorded event</p>	<ol style="list-style-type: none"> <li>1. On the left side of the screen, select the event that you want to modify.</li> <li>2. On the right side of the screen, click the <b>Body</b> and/or <b>Header</b> tab (as appropriate).</li> <li>3. Select a field.</li> <li>4. Click <b>Add Schema</b> to display the Select Schema dialog box. (For information about selecting message schemas, refer to <i>IBM Rational Integration Tester Reference Guide</i>.)</li> <li>5. Clicking <b>Finish</b> on the Select Schema dialog box prompts you to confirm that you want to apply the selected schema to the selected event.</li> </ol>
<p>Close the wizard without making any changes</p>	<p>Click <b>Cancel</b>.</p>
<p>Move to the (previous) Operation Assignment screen</p>	<p>Click <b>Back</b>.</p>
<p>Move to the (final) Summary screen of the wizard</p>	<p>Click <b>Next</b> or <b>Summary</b>.</p>

---

The Summary screen, which is the final screen of the Recorded Events wizard, enables you to review the configuration of the stub and to save the stub.



The following table describes how to use the Summary screen.

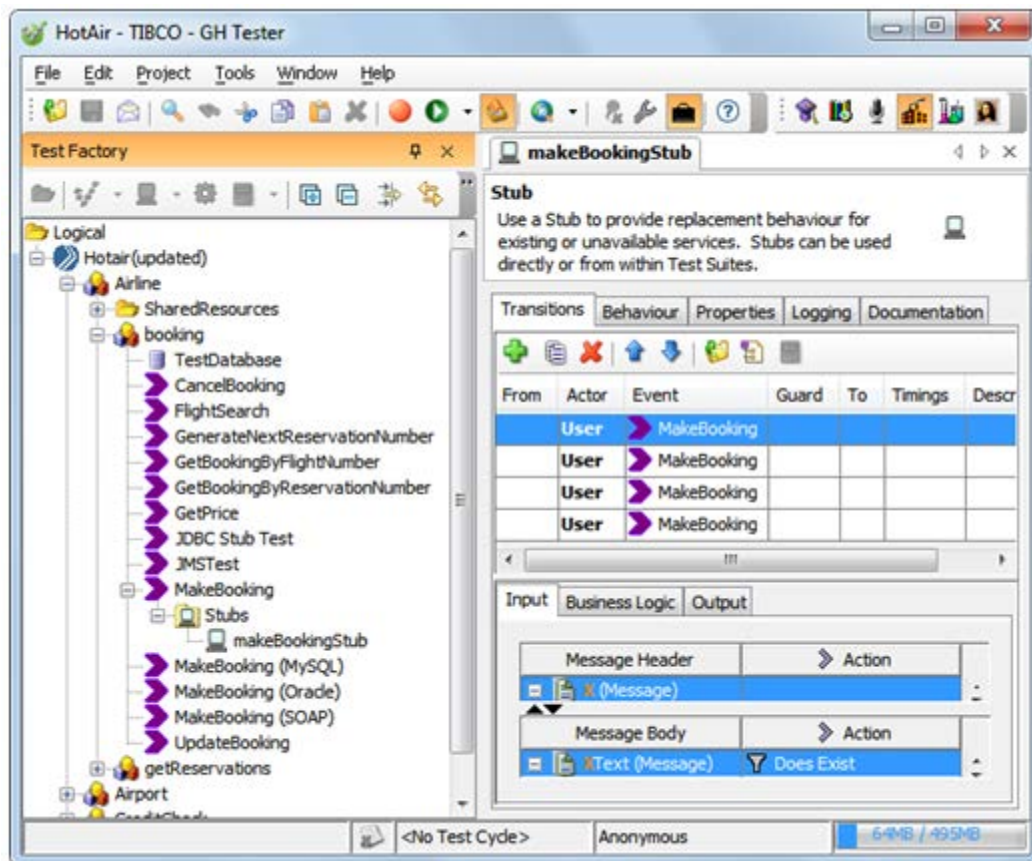
---

To...	Do this...
Open the stub in the Stub Editor after you save it and quit the Recorded Events wizard	Select the <b>Open resource after finish</b> check box.
Close the wizard without making any changes	Click <b>Cancel</b> .
Move to the (previous) Transaction Assignment screen	Click <b>Back</b> .
Save your stub	1. In the <b>Stub name</b> field, enter a name for the stub. 2. Click <b>Finish</b> .

---

After clicking **Finish** on the Summary screen:

- Rational Integration Tester's Test Factory perspective is displayed and the newly created stub is displayed under the relevant logical resource on the Test Factory perspective's component tree. If you created a stub for multiple operations, the stub is displayed under each applicable operation.
- If you selected the **Open resource after finish** check box on the Summary screen, the stub is also opened in the Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)).



---

### 3.2.3.2 Creating Data-Driven Stubs

A data-driven stub is a stub that is driven by the contents of a data source. The data fields that have been promoted to the events table (for information about this, refer to *IBM Rational Integration Tester Reference Guide*) are used to create the data source.

The following table outlines the different types of data sources that Rational Integration Tester supports. (For more information about these data source-types, refer to *IBM Rational Integration Tester Reference Guide*.)

---

Data Source-Type	Produced by Recorded Events Wizard?	Description
File data source	Yes	This reads data from a file, for example, a comma-separated value (CSV) file, a fixed width file, or some other delimited file-type.  This data can be supplied to any tests or stubs in Rational Integration Tester.
Excel data source	No	This reads data from a worksheet in a Microsoft Excel workbook file.
Database data source	No	This reads data from a table in a database or the results of a query on a database.  <b>NOTE:</b> The database must be set up in Rational Integration Tester's Architecture School perspective before this data source can be created.
Directory data source	No	This reads in a set of files, for example, a set of XML documents.

---

You can create a data-driven stub from recorded events or you can create a data source for a driven stub without using recorded events.

For example, you could create a stub by using the associated operation's Message Exchange Pattern properties (for information about this, refer to [Message Exchange Pattern \(MEP\) Method](#)) and then modifying the stub to read information from an Excel workbook file (with or without repeating elements), look up specified data, and then send any matching data in reply messages (for information about this, refer to [Modifying Message-Based Stubs](#)).

---

To create a data-driven stub from recorded events:

1. In Rational Integration Tester's Recording Studio perspective, select multiple recorded events on the Events View window.

**NOTE:** For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

2. Click the **arrow** button (↵) next to the **Save Stub from selected events** button (📄) on the Events View toolbar and then click **Save Parameterized Stub** on the shortcut menu.

Alternatively, right-click the messages and click **Save Parameterized Stub** on the shortcut menu.

The Operation Assignment screen of the Recorded Events wizard is displayed.

The Operation Assignment screen, which is the third screen of the wizard, enables you to modify (if you wish) the operation and the events associated with the stub.

**NOTE:** The first and second screens of the Recorded Events wizard are the Resource Type and Data Storage screens. The screens are displayed only if you click the **Save** button (📄) or press CTRL+S. Clicking the **Save Stub from selected events** button (📄) bypasses the Resource Type and Data Storage screens.

For information about using Operation Assignment screen, refer to [Creating Basic Stubs](#).

**NOTE:** If you are creating a stub for multiple operations, Rational Integration Tester will attempt to verify that all selected recorded events are associated with the correct operations.

3. Clicking **Next** on the Operation Assignment screen displays the Transaction Assignment screen.

The Transaction Assignment screen, which is the fourth screen of the Recorded Events, enables you to group events into transactions.

**NOTE:** The groups that you choose are important because they will determine the number of merged messages and thus the number of remaining screens in the Recorded Events wizard. For example, one recorded event may have a single transaction, which means that there is only one mapping. Alternatively, two recorded events may be

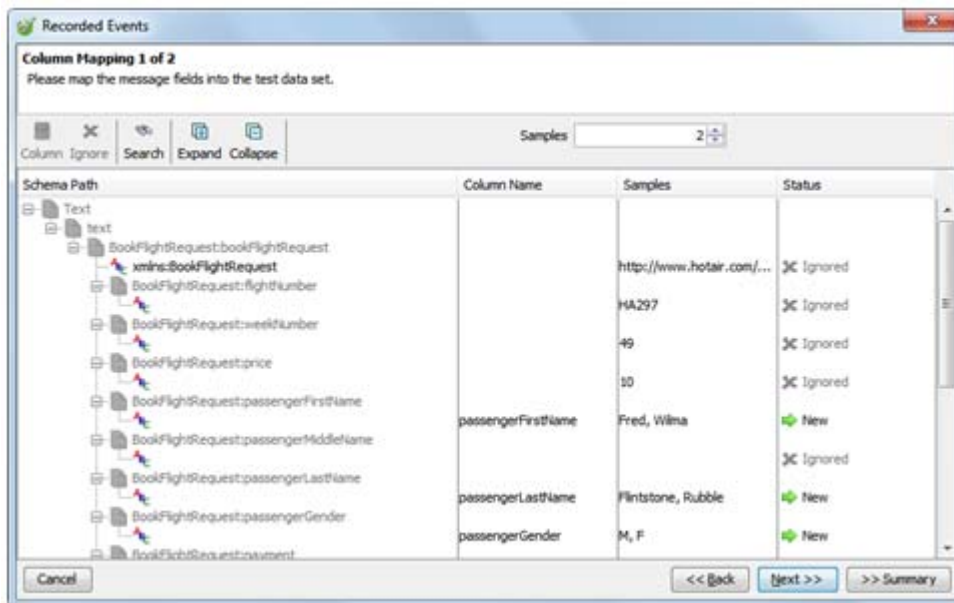
---

grouped into two groups, which means that there are two transactions and thus only one mapping. Equally, two recorded events may be put into a single group, which means that there are two mappings.

For information about using Transaction Assignment screen, refer to [Creating Basic Stubs](#).

4. Clicking **Next** on the Transaction Assignment screen displays the Column Mapping screen.

The Column Mapping screen enables you to map the fields in the selected recorded events to the data source that will be used for data-driven stubbing.



The following table describes how to use the Column Mapping screen.

---

To...	Do this...
Specify that a field in a recorded event will be a column in the data source	<ol style="list-style-type: none"><li>1. Select the field.</li><li>2. Click <b>Column</b>. The Map to column dialog box is displayed.</li><li>3. Click <b>OK</b>. For the selected field, New is displayed under <b>Status</b>.</li></ol>

---

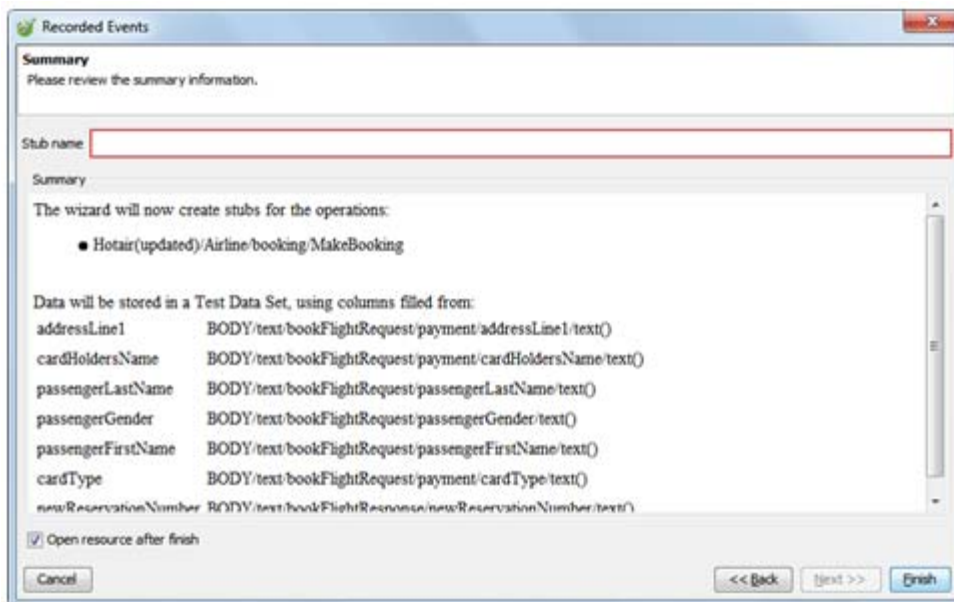
---

---

To...	Do this...
Map a field in a recorded event to an existing column	<ol style="list-style-type: none"><li>1. Select the field.</li><li>2. Click <b>Column</b>. The Map to column dialog box is displayed.</li><li>3. In the list, click the column name to which you want to map the field.</li><li>4. Click <b>OK</b>.</li></ol>
Specify that a field in a recorded event is not mapped	<ol style="list-style-type: none"><li>1. Select the field.</li><li>2. Click <b>Ignore</b>.</li></ol>
Search for a specific data item in a recorded event	<ol style="list-style-type: none"><li>1. Click <b>Search</b>.</li><li>2. In the <b>Find</b> field (displayed on the lower half of the screen), enter your search query.</li><li>3. Press ENTER.</li></ol>
Increase or decrease the number of example values displayed for each field	In the <b>Samples</b> box, enter or select the number of examples you want to display (default value: 2).
Expand or collapse a node	<ol style="list-style-type: none"><li>1. Click the node.</li><li>2. Click <b>Expand</b> or <b>Collapse</b> (as applicable).</li></ol>
Close the wizard without making any changes	Click <b>Cancel</b> .
Move to the (previous) Transaction Assignment screen	Click <b>Back</b> .
Move to the next screen	Click <b>Next</b> . <b>NOTE:</b> The next screen may be another Data Mapping screen because the number of Column Mapping screens displayed by the Recorded Events wizard depends on the number of merged messages (which determined by the using the Transaction Assignment screen).
Move to the (final) Summary screen of the wizard	Click <b>Summary</b> .

---

The Summary screen, which is the final screen of the Recorded Events wizard, enables you to review the configuration of the stub and to save the stub.

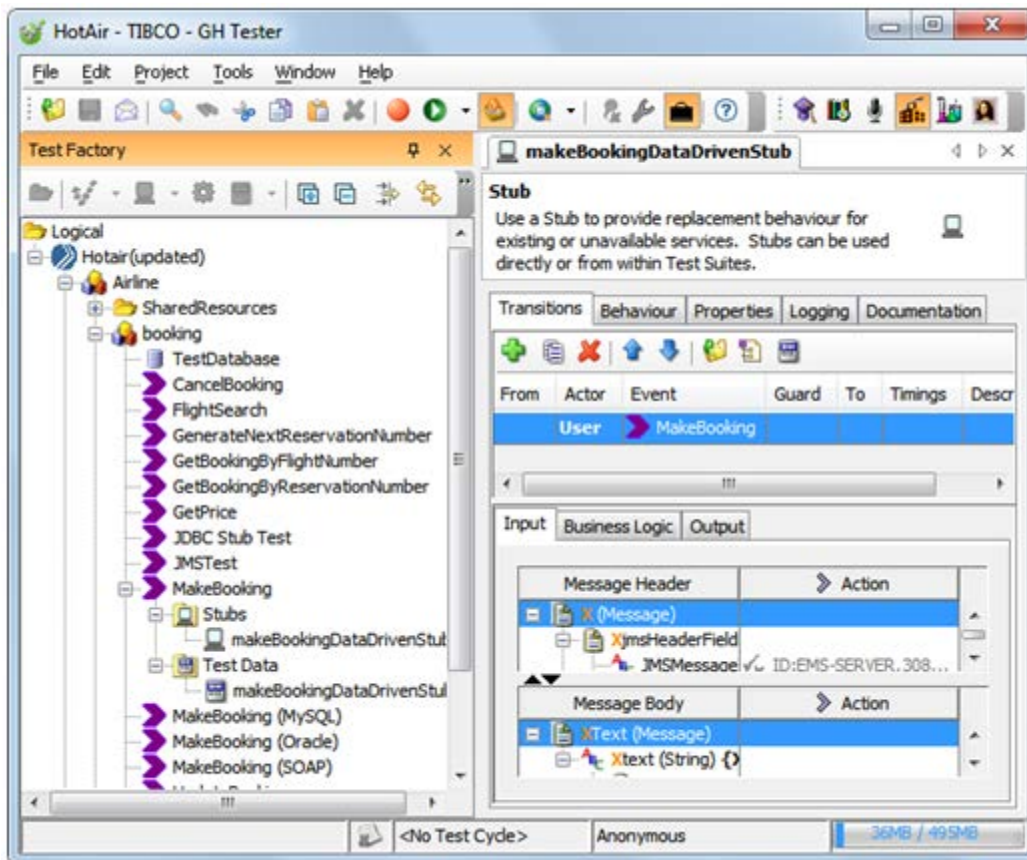


For information about using the Summary screen, refer to [Creating Basic Stubs](#).

After clicking **Finish** on the Summary screen:

- Rational Integration Tester's Test Factory perspective is displayed, and the newly created stub and data source are displayed under the relevant logical resource on the Test Factory perspective's component tree. If you created a stub for multiple operations, the stub is displayed under each applicable operation.
- If you selected the **Open resource after finish** check box on the Summary screen, the stub is also opened in the Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)). A Lookup Test Data action that uses the new data source is displayed on the **Business Logic** tab on the Transitions tab of the Stub Editor.





**NOTE:** The **Output** tab may not detect a Send Reply action on the **Business Logic** tab if the Send Reply action is nested under a Lookup Test Data action.

---

### 3.2.3.3 Creating Data Model-Driven Stubs

Basic (hard-coded) and parameterized (data-driven) stubs are examples of simple stub-types because the data that is provided when the stub is run is either hard-coded in the stub itself or is taken from a simple data source (whichever is applicable). In each case, the same data is used each time when the stub is started.

However, the testing of complex systems may require persistent storage of data so that a stub can save information across multiple operations. This enables you to virtualize services that create data, for example, a customer, and then use that data in a “GetCustomer” service without having to know a design time the data will be used. It is this capability that distinguishes simple stubs from those that are coordinated to provide a virtualized application.

In cases where you require multiple operations within a stub, or multiple stubs to share data to provide a virtualized application, you should use a data model to hold such data. Data models can be shared across stubs. In addition, data models can hold data about multiple entities and their relationships, such as customers and orders.

Therefore, a data model is a simple view of entities and their relationships that can be used by stubs to persist information. Data models exist as assets within a project. A stub’s properties define which data model it is using (if any). Stubs can read and modify data held in a data model. The Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)) enables you to indicate whether an operation is trying to create, update, or delete data based on the received message, reducing the time and effort required to create rich stubs.

In Rational Integration Tester, you can create a data model by using the Data Model Editor (for information about this, refer to [Appendix A: Using the Data Model Editor](#)). Alternatively, you can create a data model while creating a data model-driven stub from recorded events. This second method analyzes the message in recorded events and creates a data model and entities within that data model based on the messages. Operations that are created will connected to the data model automatically.

To create a data model-driven stub from recorded events:

1. In Rational Integration Tester’s Recording Studio perspective, select multiple recorded events on the Events View window.

**NOTE:** For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.



2. Click the **arrow** button (↵) next to the **Save Stub from selected events** button (📁) on the Events View toolbar and then click **Save Data Model Stub** on the shortcut menu.

---

Alternatively, right-click the messages and click **Save Data Model Stub** on the shortcut menu.

The Operation Assignment screen of the Recorded Events wizard is displayed.

The Operation Assignment screen, which is the third screen of the wizard, enables you to modify (if you wish) the operation and the events associated with the stub.

**NOTE:** The first and second screens of the Recorded Events wizard are the Resource Type and Data Storage screens. The screens are displayed only if you click the **Save** button (  ) or press CTRL+S. Clicking the **Save Stub from selected events** button (  ) bypasses the Resource Type and Data Storage screens.

For information about using Operation Assignment screen, refer to [Creating Basic Stubs](#).

**NOTE:** If you are creating a stub for multiple operations, Rational Integration Tester will attempt to verify that all selected recorded events are associated with the correct operations.

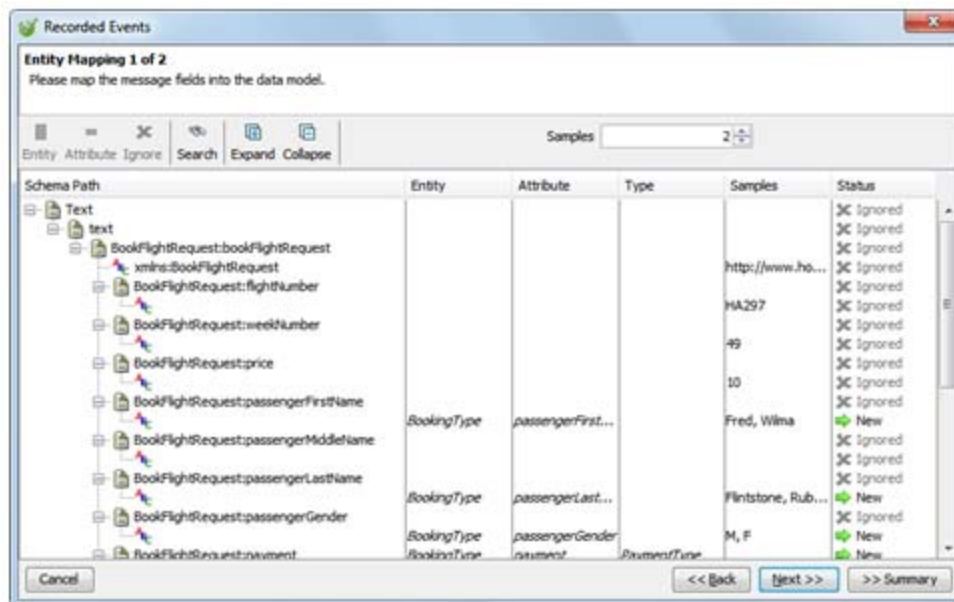
3. Clicking **Next** on the Operation Assignment screen displays the Transaction Assignment screen.

The Transaction Assignment screen, which is the fourth screen of the Recorded Events, enables you to group events into transactions.

**NOTE:** The groups that you choose are important because they will determine the number of merged messages and thus the number of remaining screens in the Recorded Events wizard. For example, one recorded event may have a single transaction, which means that there is only one mapping. Alternatively, two recorded events may be grouped into two groups, which means that there are two transactions and thus only one mapping. Equally, two recorded events may be put into a single group, which means that there are two mappings.

For information about using Transaction Assignment screen, refer to [Creating Basic Stubs](#).

4. Clicking **Next** on the Transaction Assignment screen displays the Entity Mapping screen.



The Entity Mapping screen enables you to map the fields in the selected recorded events to the data source that will be used for data model-driven stubbing.

The following table describes how to use the Entity Mapping screen.

To...	Do this...
Specify that a field in a recorded event will be an entity in the data model	<ol style="list-style-type: none"> <li>1. Select the field.</li> <li>2. Click <b>Entity</b>. The Map to entity dialog box is displayed.</li> <li>3. Click <b>OK</b>. For the selected field, New is displayed under <b>Status</b>.</li> </ol>
Specify that a field in a recorded event will be an attribute of a particular entity in the data model	<ol style="list-style-type: none"> <li>1. Select the field.</li> <li>2. Click <b>Attribute</b>. The Map to Attribute dialog box is displayed.</li> <li>3. Click <b>OK</b>. For the selected field, New is displayed under <b>Status</b>.</li> </ol>
Specify that a field in a recorded event is not included in the data model	<ol style="list-style-type: none"> <li>1. Select the field.</li> <li>2. Click <b>Ignore</b>.</li> </ol>
Search for a specific data item in a recorded event	<ol style="list-style-type: none"> <li>1. Click <b>Search</b>.</li> <li>2. In the <b>Find</b> field (displayed on the lower half of the screen), enter your search query.</li> <li>3. Press ENTER.</li> </ol>

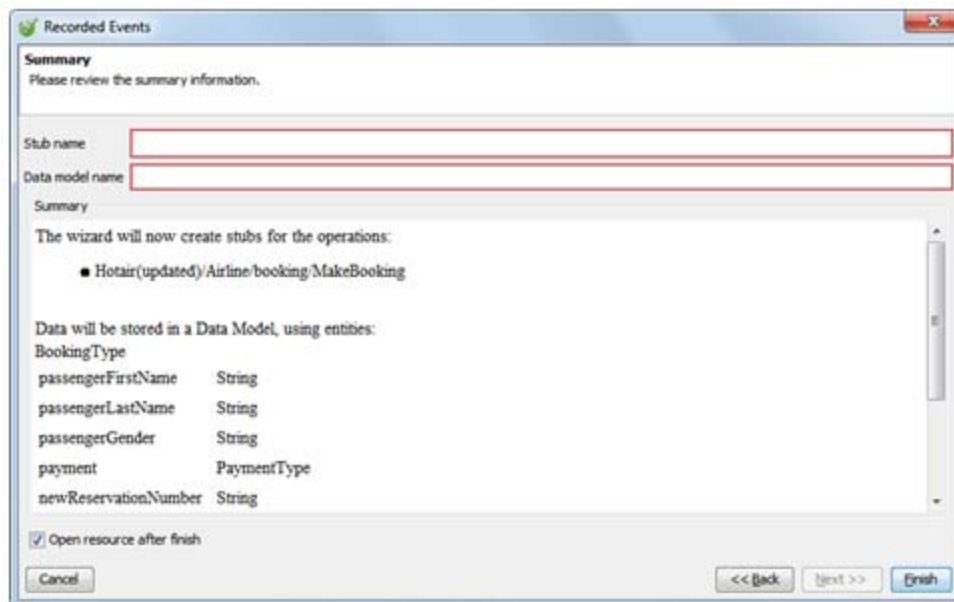
---

---

To...	Do this...
Increase or decrease the number of example values displayed for each field	In the <b>Samples</b> box, enter or select the number of examples you want to display (default value: 2).
Specify that a field is a “key” field so that it can be used to match data between different messages	Under the <b>Key</b> column, select the check box of the field.
Expand or collapse a node	1. Click the node. 2. Click <b>Expand</b> or <b>Collapse</b> (as applicable).
Close the wizard without making any changes	Click <b>Cancel</b> .
Move to the (previous) Transaction Assignment screen	Click <b>Back</b> .
Move to the next screen	Click <b>Next</b> . <b>NOTE:</b> The next screen may be another Entity Mapping screen because the number of Entity Mapping screens displayed by the Recorded Events wizard depends on the number of merged messages (which determined by the using the Transaction Assignment screen).
Move to the (final) Summary screen of the wizard	Click <b>Summary</b> .

---

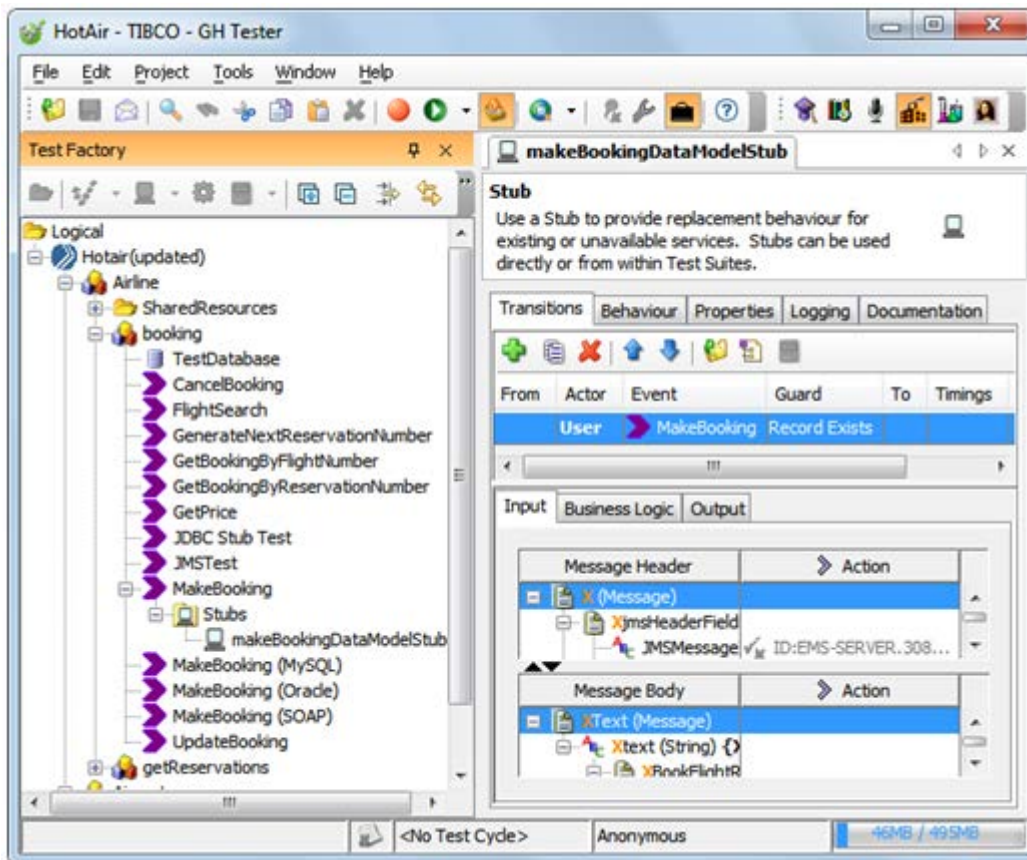
The Summary screen, which is the final screen of the Recorded Events wizard, enables you to review the configuration of the stub and to save the stub and the data model.



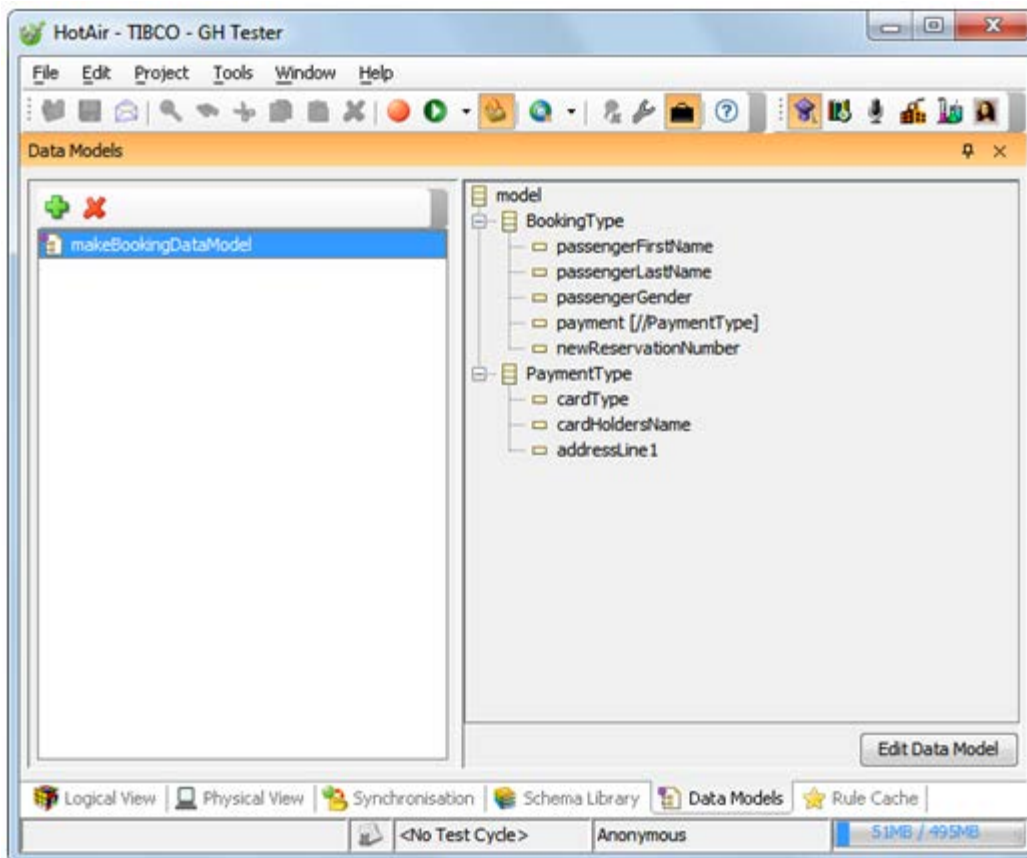
For information about using the Summary screen, refer to [Creating Basic Stubs](#).

After clicking **Finish** on the Summary screen:

- Rational Integration Tester's Test Factory perspective is displayed, and the newly created stub is displayed under the relevant logical resource on the Test Factory perspective's component tree. If you created a stub for multiple operations, the stub is displayed under each applicable operation.
- If you selected the **Open resource after finish** check box on the Summary screen, the stub is also opened in the Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)).



- If you view the Architecture School perspective's Data Models window, the newly created data model is listed on the upper left of the window and the data model's details can be displayed on Data Models window by selecting it. For information about editing a data model, refer to [Editing Data Models](#).





---

#### 3.2.3.4 Creating Deterministic Stubs

A deterministic stub comprises a series of Receive Request/Send Response or Subscribe/Publish actions that are based on the message used to create the stub.

To create a deterministic stub from recorded events:

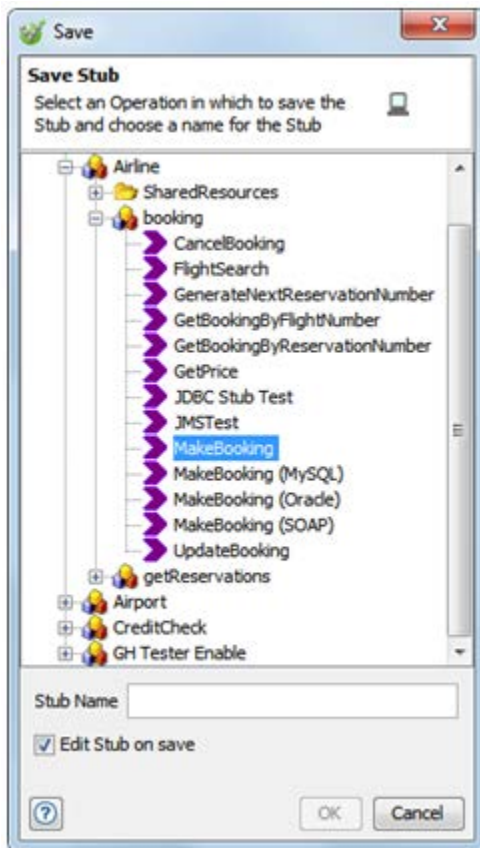
1. In Rational Integration Tester's Recording Studio perspective, select multiple recorded events on the Events View window.

**NOTE:** For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

2. Click the **arrow** button (↘) next to the **Save Stub from selected events** button (📁) on the Events View toolbar and then click **Save Deterministic Stub** on the shortcut menu.

Alternatively, right-click the messages and click **Save Deterministic Stub** on the shortcut menu.

The Save Stub dialog box is displayed.

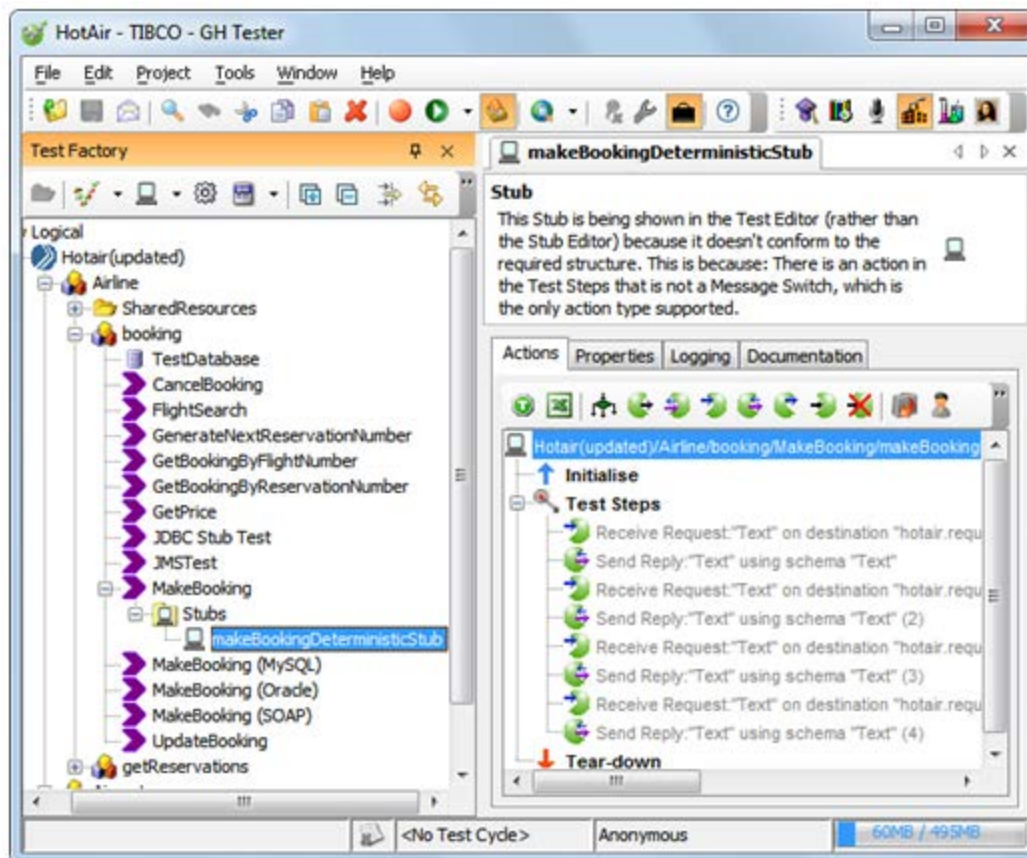


3. Select the operation with which you want to associate the stub.

**NOTE:** If there is only one operation associated with the selected recorded events, it is selected automatically when the Save Stub dialog box is displayed.

4. In the **Stub Name** field, enter the name of the stub.
5. **Optional:** Clear the **Edit Stub on save** check box if you do not want to edit the stub after creating it.
6. Click **OK**.

If the **Edit Stub on save** check box on the Save Stub dialog box was selected, Rational Integration Tester's Test Factory perspective is displayed, and the newly created stub is displayed under the relevant logical resource on the Test Factory perspective's component tree and the stub is displayed on a basic Stub Editor screen that does not support transitions and behaviours (for information about modifying stub, refer to [Modifying Message-Based Stubs](#)).



If the **Edit Stub on save** check box is cleared and you want to verify that the stub has been created, you must open Rational Integration Tester's Test Factory's perspective and view the component tree.

---

## 3.3 Modifying Message-Based Stubs

In Rational Integration Tester 5.4.0 (or later), you can use either the Test Editor or the Stub Editor to create, modify, and enhance any legacy or new stubs created by Rational Integration Tester.

For example, you may want to create an advanced message-based stub that can record and report its internal state so that you can create a virtualized application that replicates a complex live transaction processing system.

The following table outlines how to open the Test Editor and the Stub Editor.

If you want to...	Do this...	Then...
Modify a stub created by Rational Integration Tester 5.2.11 (or earlier).	In Rational Integration Tester's Test Factory perspective: <ul style="list-style-type: none"><li>• Double-click the stub.</li><li>• Alternatively, right-click the stub and click <b>Open</b> on the shortcut menu.</li></ul>	The selected stub is displayed in the Stub Editor if the stub does not contain any features that cannot be handled by the Stub Editor. Otherwise, the selected stub is displayed in the Test Editor.
Modify a stub created by Rational Integration Tester 5.4.0 (or later).	In Rational Integration Tester's Test Factory perspective: <ul style="list-style-type: none"><li>• Double-click the stub.</li><li>• Alternatively, right-click the stub and click <b>Open</b> on the shortcut menu.</li></ul>	The selected stub is displayed in the Stub Editor. <b>NOTE:</b> You can also modify a stub while it is running. For information about this, refer to <a href="#">Modifying Running Stubs</a> .
Use Test Editor to modify a stub created by any release of Rational Integration Tester.	In Rational Integration Tester's Test Factory perspective, right-click the stub and click <b>Open With &gt; Test Editor</b> on the shortcut menu.	The selected stub is displayed in the Test Editor.

The Stub Editor comprises the following tabs:

- **Transitions**
- **Behaviour**
- **Properties**
- **Logging**
- **Documentation**

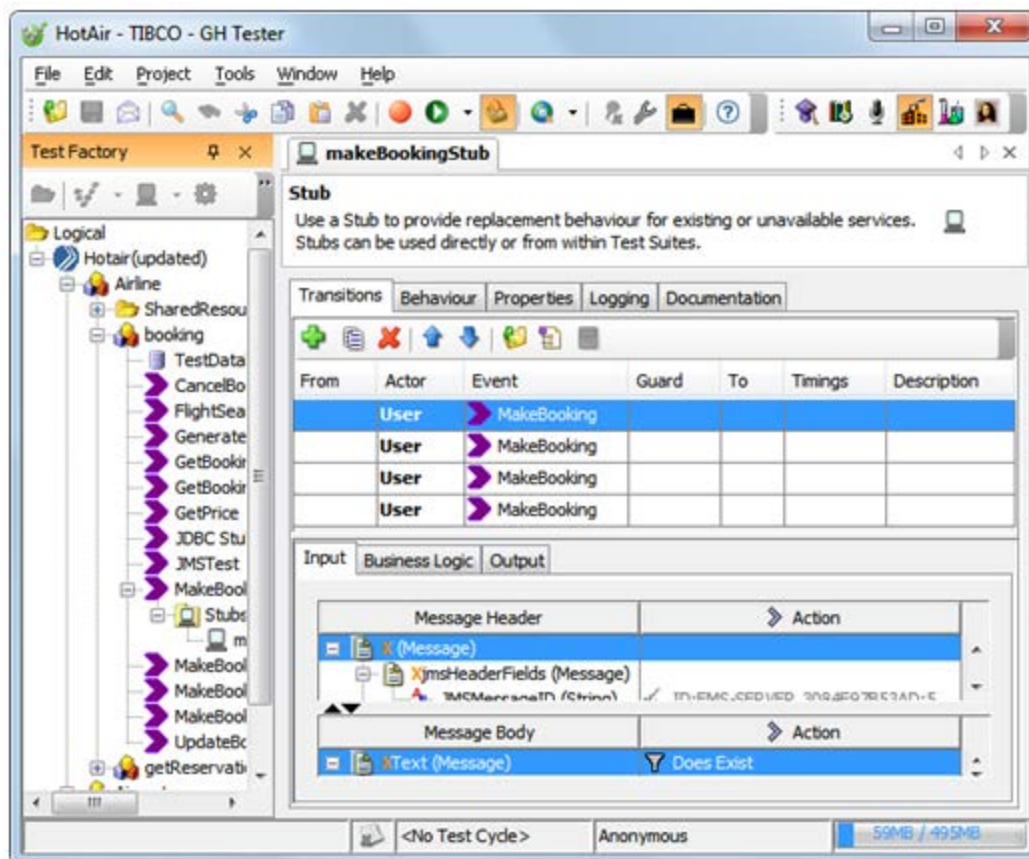
---

The following sections describe how to use these tabs.

**NOTE:** For information about using the Test Editor, refer to *IBM Rational Integration Tester Reference Guide*.

### 3.3.1 Transitions Tab

This tab enables you to define how the stub will handle internally and/or externally received events. Strictly defined, a *transition* refers to a transition between two states. However, quite often, stubs will not use states, so a transition can be regarded as the action that a stub will take when a message arrives on the operation for which you want to create a stub.



**NOTE:** This tab is not displayed when editing a deterministic stub.

The following sections describe how to use the buttons, lists table, and three tabs on the **Transitions** tab.

---

### 3.3.1.1 Buttons and List Table

The following table describes each button on the upper half of the **Transitions** tab.

Button	Description
Add Transition	Clicking this button creates a new transition for the currently selected operation.
Clone Transition	Clicking this button creates a copy of the currently selected transition in the list table.
Delete Transition	Clicking this button deletes the currently selected transition in the list table.
Move Transition Up/ Down	Clicking these buttons changes the list order of the currently selected transition in the list table and consequently the order in which any guard conditions of any transitions listed in the table are evaluated while the stub is being executed.
Open operation referenced by transition	Clicking this button opens the associated operation's Operation dialog box and changes the perspective from Test Factory to Architecture School.
Open data model referenced by stub	Clicking this button opens the data model (a comma-separated value (CSV) file), if any, associated with the stub. <b>NOTE:</b> You must use the <b>Properties</b> tab to associate a data model with the stub. (For information about using the <b>Properties</b> tab, refer to <a href="#">Properties Tab</a> .)
Open data used by selection	Clicking this button opens any data associated with the currently selected transition in the list table.

The list table on the upper half of the **Transitions** tab lists all the transitions contained in the stub.

When a stub receives an event, it searches the list of transitions from top to bottom to find a transition that has been set up to handle the event received, based on the stub's current state. You can use message filters and guards if you want to ensure that the contents of a message determines whether and how a transition is processed.

Many stubs will not use more than one state, so the **From** and **To** fields may be left blank. If so, all transitions for such stubs will keep those stubs in a single state.

---

The following table describes the fields in the list table.

Field	Entry	Description
From	Optional	<p>This field enables you to specify an initial state for an event.</p> <p>For example, if you are building a stub that can track user sessions, you might want to have an initial state called <code>LoggedOut</code> for an operation <code>Login</code> operation. (The final state for the corresponding <code>Logout</code> operation might be <code>LoggedIn</code>.)</p> <p>Clicking the field displays a drop-down list that lists any states already defined for the stub and an <b>Add new...</b> option that enables you to create a new state.</p> <p>Clicking <b>Add new...</b> in the <b>From</b> list opens the New State dialog box. In the New State dialog box, you must enter a name for the new state in the <b>Name</b> field. You can enter a description for the new state in the <b>Description</b> field but that is optional. Clicking <b>OK</b> closes the New State dialog box.</p> <p>If this field and the <b>To</b> field are left blank, Rational Integration Tester will assume that this transition will work for any state of the currently selected stub, so the stub will be stateless.</p> <p>You can also create a new state by clicking <b>New</b> next to the <b>States</b> field on the <b>Properties</b> tab. (The <b>Edit</b> and <b>Delete</b> buttons next to the States field enable you to edit and delete states as required.)</p> <p><b>NOTE:</b> The <b>From</b> field becomes editable only after an event has been selected for the transition.</p>
Actor	Mandatory	<p>This field enables you to specify an entity that causes an event to happen.</p> <p>When creating a new transition, clicking this field displays a drop-down list that includes <b>User</b> (that is, a user of the service) and any behavioural entities defined on the <b>Behaviour</b> tab. (For information about using the <b>Behaviour</b> tab, refer to <a href="#">Behaviour Tab</a>.) For most transitions, <b>Actor</b> will be set to <b>User</b>.</p> <p><b>NOTE:</b> This field is not editable after an event has been specified (by using the <b>Event</b> field) for the transition.</p>

---

Field	Entry	Description
Event	Mandatory	<p>In computing generally, an “event” is something that happens that affects the system. In Rational Test Virtualization Server, an event refers to an operation.</p> <p>This field enables you to specify the operation associated with the currently selected transition.</p> <p>When creating a new transition, clicking this field displays a drop-down list that includes any operations already associated with the stub and a <b>Browse...</b> option that enables you to select a different operation.</p> <p>Clicking <b>Browse...</b> in the <b>Event</b> list opens the Select an Operation dialog box. In the Select an Operation dialog box, select an operation from your project’s logical resources and click <b>OK</b>.</p> <p><b>NOTE:</b> This field is not editable until after an event has been selected for the transition.</p>
Guard	Optional	<p>This field enables you to specify conditions that will determine if an event will be handled. For example, you could create a guard based on the data held in a data model associated with the currently selected stub.</p> <p>Rational Integration Tester tags can be used to create guard conditions. (For information about creating and using tags, refer to <i>IBM Rational Integration Tester Reference Guide</i>.)</p> <p><b>NOTE:</b> You must use the <b>Input</b> tab on the lower half of the <b>Transitions</b> tab to create, modify, and specify the guard condition of a transition. (For information about using the <b>Input</b> tab, refer to <a href="#">Input Tab</a>.)</p> <p><b>NOTE:</b> This field is not available until after an event has been specified for the currently selected transition.</p>

---



---

Field	Entry	Description
To	Optional	<p>This field enables you to specify a final state for an event. For example, if you are building a stub that can track user sessions, you might want to have a final state called <code>LoggedIn</code> for an operation called <code>Login</code>. (The final state for the corresponding <code>Logout</code> operation might be <code>LoggedOut</code>.)</p> <p>Clicking the field displays a drop-down list that lists any states already defined for the stub and an <b>Add new...</b> option that enables you to create a new state.</p> <p>Clicking <b>Add new...</b> in the <b>To</b> list opens the New State dialog box. In the New State dialog box, you must enter a name for the new state in the <b>Name</b> field. You can enter a description for the new state in the <b>Description</b> field but that is optional. Clicking <b>OK</b> closes the New State dialog box.</p> <p>If this field and the <b>From</b> field are left blank, Rational Integration Tester will assume that this transition will work for any state of the currently selected stub, so the stub will be stateless.</p> <p>You can also create a new state by clicking <b>New</b> next to the <b>States</b> field on the <b>Properties</b> tab. (The <b>Edit</b> and <b>Delete</b> buttons next to the States field enable you to edit and delete states as required.)</p> <p><b>NOTE:</b> The <b>To</b> field becomes editable only after an event has been selected for the transition.</p>
Timings	Optional	<p>This field enables you to specify a timing override for the currently selected transition. If a timing override is set for a specific transition, it will override any response times set for the stub. (For information about setting the response times of a stub, refer to <a href="#">Behaviour Tab</a>.)</p> <p>Double-clicking this field opens the Timing Override dialog box. In the Timing Override dialog box, select an option in the <b>Delay Distribution</b> list, enter minimum and maximum delays (in milliseconds) in the fields provided, and click <b>OK</b>. Clicking <b>Cancel</b> or <b>Clear</b> closes the dialog box without saving any changes.</p> <p><b>NOTE:</b> This field is not available until after an event has been specified for the transition.</p>

---

---

Field	Entry	Description
Description	Optional	<p>This field enables you to enter a narrative description for the currently selected transition.</p> <p>To enter a description for a transition, select it and double-click this field.</p> <p><b>NOTE:</b> This field is not editable until after an event has been specified for the transition.</p>

---

### 3.3.1.2 Input Tab

For each transition, this tab enables you to specify:

- A guard condition (optional).
- Incoming message field actions (optional).

For example, if you have created a basic (hard-coded) stubs from recorded events (for information about this, refer to [Creating Basic Stubs](#)), the stub will filter out any messages that do not match exactly the message(s) received while recording the events used to build the stub, so the stub will not send a response message.

However, if you want that basic stub to respond to any messages that match the message structure in the stub irrespective of the values within the fields of those messages, you can use the **Input** tab to disable any filtering that checks for exact field matches and thus make your stub more versatile.

The tab comprises four option buttons for specifying and configuring a guard condition for a transition, and two windows for setting incoming message field actions (for a transition).

The following sections describe how to use these controls.

#### Guard Conditions

Guard conditions are Boolean expressions that affect the behaviour of a stub by enabling actions or transitions only when they evaluate to “TRUE” and disabling them when they evaluate to “FALSE”.

---

The following table describes the **Guard** option buttons on the **Input** tab on the **Transitions** tab.

Option Button	Description
None	<p>Click this option button (it is also the default) if you do not want to specify any guard conditions for the currently selected transition.</p> <p>The <b>Guard</b> field on the list table will be blank if this option button clicked or if none of the <b>Guard</b> option buttons is clicked.</p>
Record Exists	<p>Click this option button if a data model exists <b>and</b> you want to run the business logic/transition if the record referenced by the incoming message exists.</p> <p>If this option button clicked, <code>Record Exists</code> will be displayed under <b>Guard</b> on the list table for the currently selected transition.</p> <p>For a data model-driven message-based stub, <code>Record Exists</code> causes the stub to use mappings (from message contents to data model tag names) defined in the Message Editor when looking up records in the data model.</p> <p>If a record already exists, the <code>Record Exists</code> guard passes. If a record does not already exist, the <code>Record Doesn't Exist</code> guard passes.</p> <p>This enables you to have two transitions for an incoming message and to respond differently depending on whether the specified data is found.</p> <p>For example, if the service is "Find Customer" and the "Customer ID" field in the message is key field, customer data or an error message can be returned by using the two transitions and the guards without having to write the business logic.</p>
Record Doesn't Exist	<p>Click this option button if a data model exists <b>and</b> you want to run the business logic/transition if the record in the data model does <b>not</b> exist.</p> <p>If this option button clicked, <code>Record Doesn't Exist</code> will be displayed under <b>Guard</b> on the list table for the currently selected transition.</p>

---

---

Option Button	Description
Other	<p>Click this button if you want to specify a custom Boolean expression for the currently selected transition.</p> <p>If this option button is clicked a drop-down list, a text box, and buttons are displayed. The controls enable you to create and test your own Boolean expressions.</p> <p>The drop-down list contains the following options:</p> <ul style="list-style-type: none"><li>• <b>ECMAScript</b> (default; for example, JavaScript)</li><li>• <b>Legacy</b></li></ul> <p>Clicking <b>ECMAScript</b> displays a <b>Test</b> button. Clicking <b>Test</b> validates your custom Boolean expression and displays an Output dialog box that lists any errors.</p> <p>Clicking <b>Legacy</b> displays <b>Add</b>, <b>Delete</b>, and <b>Test</b> buttons and an <b>'OR' Expressions</b> check box. Clicking <b>Add</b> makes the text box available. Clicking <b>Test</b> validates your custom Boolean expression and displays an Output dialog box that lists any errors.</p> <p>If this option button clicked and a custom Boolean expression is entered, the starting characters of the expression will be displayed under <b>Guard</b> on the list table for the currently selected transition.</p>
No Match	<p>Click this option button if you want a transition's specified event (operation) to change state if a matching record is not found.</p> <p>If this option button clicked, <b>No Match</b> will be displayed under <b>Guard</b> on the list table for the currently selected transition.</p>

---

### Incoming Message Field Actions

The Message Header and Message Body windows on the **Input** tab on the **Transitions** tab enable you to specify actions for each field in the incoming message.

Double-clicking any field on the windows opens the Field Editor dialog box. The **Action** column is a single view of the action groups (Filter, Validate, and Store) for incoming messages.

For example, if you want to disable any filtering for exact field matches:

1. Select all of the fields on the **Input** tab.
2. Right-click the selected fields and click **Contents > Field Actions > Filter > Equality** on the shortcut menus.

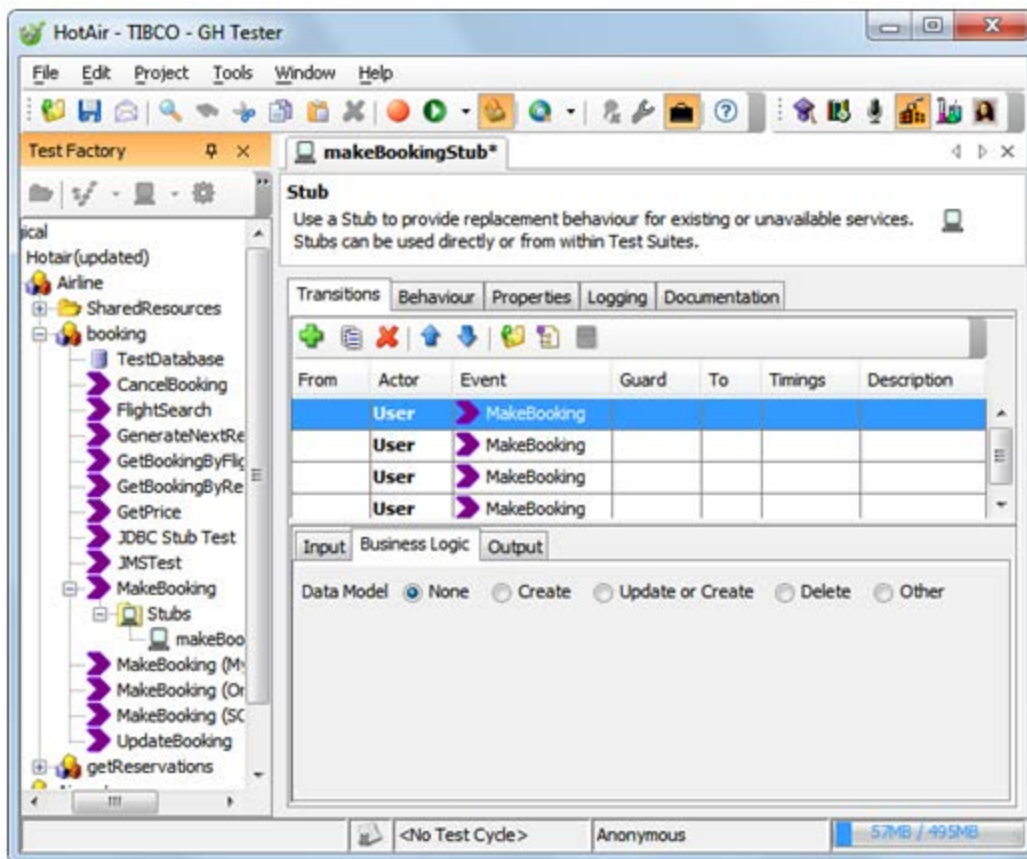
Equality checking for the selected fields is disabled and fewer filter icons are displayed on the **Input** tab.

---

**NOTE:** Double-clicking a field opens the Field Editor dialog box. (For information about using the Field Editor dialog box, refer to [Field Editor](#).)

### 3.3.1.3 Business Logic Tab

This tab enables you to specify what actions the stub will take when it executes a transition. For data model stubs, it is possible to configure the stub to create, read, update or delete (CRUD) automatically from the associated data model.



If you are not using a data model stub or simple hard-coded stub, you will need to write business logic using actions in the same way that you would for writing a test.

A parameterized stub that uses data sets will need a Lookup action (or similar) in its business logic to retrieve the data. If you created a stub with a data set from the wizard, these actions will have been added automatically into the business logic by the wizard.

---

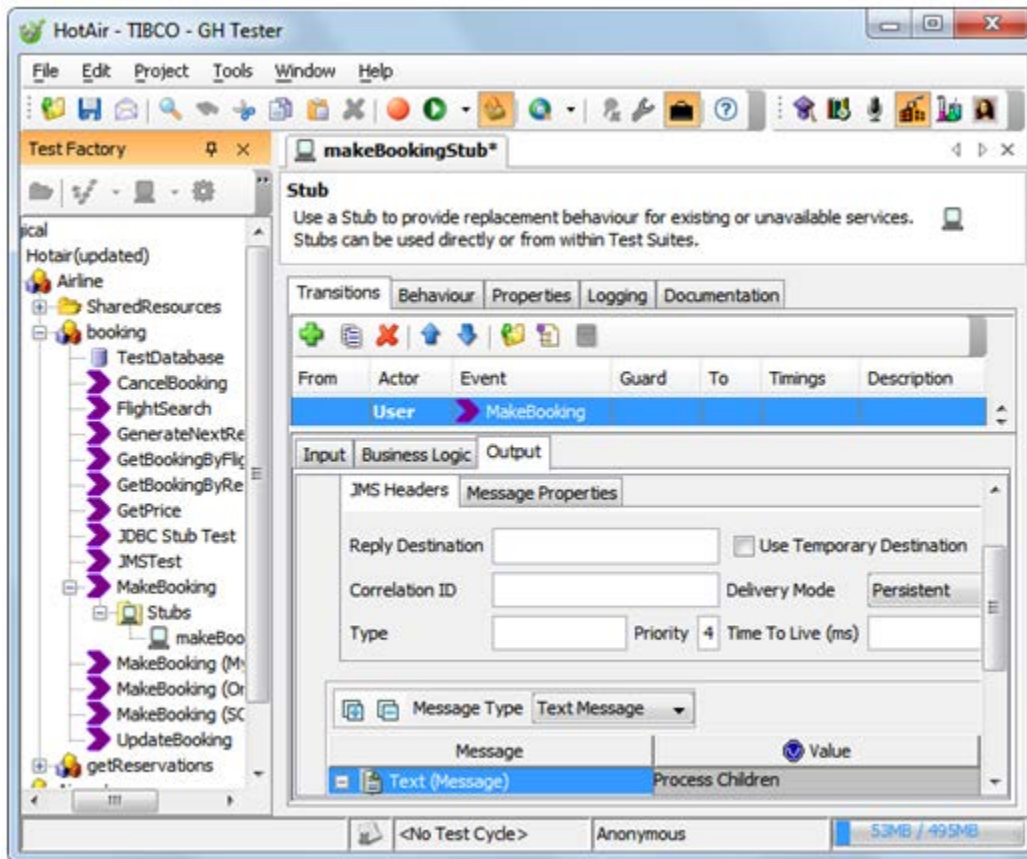
The following table describes the five option buttons on the **Business Logic** tab on the **Transitions** tab.

Option Button	Description
None	Click this option button (it is also the default) if you do not want to specify any processing rules for any incoming messages for the currently selected transition.
Create	Click this option button if the stub has an associated data model and you want the result of this operation to be a new record in the data model based on the content of the incoming message.
Update or Create	Click this option button if the stub has an associated data model and you want the result of this operation to be a new or updated record in the data model based on the content of the incoming message.
Delete	Click this option button if the stub has an associated data model and you want the result of this operation to be a deleted record in the data model based on the content of the incoming message.
Other	<p>Click this option button to create custom processing rules for any incoming messages for the currently selected transition.</p> <p>Clicking this option button opens a Test Editor window on the <b>Business Logic</b> tab.</p> <p>The Test Editor window enables you to define actions that will be executed while processing any incoming messages. (For information about using the Test Editor, refer to <i>IBM Rational Integration Tester Reference Guide</i>.)</p>

---

### 3.3.1.4 Output Tab

For each transition, this optional tab enables you to specify outgoing (reply) message field values.



The tab comprises a **Send Response** check box, a reply header window, and a reply message (body) window.

Select the **Send Response** check box if you want a reply message to be sent in response to each processed incoming message.

The following sections describe how to use the windows on the tab.

#### Reply Header Window

The Reply Header window is on the upper half of the **Output** tab on the **Transitions** tab. The window enables you to override transport-specific response settings and header metadata of the reply message for the currently selected transition.

For information about editing message headers, refer to *IBM Rational Integration Tester Reference Guide*.

---

## Reply Message Window

The Reply Message window is on the lower half of the **Output** tab on the **Transitions** tab. The window enables you to view and configure the validation of the reply message for the currently selected transition.

Depending on the transport associated with the operation that is associated with the currently selected transition, the message-type can be selected by using the **Message Type** list above the message body.

The body of the reply message is displayed in a Publisher/Subscriber view that includes enabled values, which enables you to select or clear the **Enabled Value** check box of each element/field, thus enabling or disabling the value specified for the content of that element/field.

Double-clicking an field on the reply message window opens the Field Editor dialog box. The **Value** column is a single view of the action groups (Value, Validate, and Store) for reply messages. (For information about using the Field Editor dialog box, refer to [Field Editor](#).)

### 3.3.1.5 Field Editor

The Field Editor dialog box enables you to specify how a message field is populated.

It comprises three tabs (action groups) but the tabs displayed depend on message-type.

---

Field Editor Tabs for Incoming Messages	Field Editor Tabs for Outgoing Messages
Filter	Value
Validate	Validate
Store	Store

---

To open the Field Editor dialog box:

- Double-click a field or element name on the **Input** or **Output** tabs on the **Transitions** tab of the Stub Editor.
- Alternatively, right-click a field on the **Input** or **Output** tabs on the **Transitions** tab of the Stub Editor and click **Contents > Edit** on the shortcut menus.

**NOTE:** The performance of the Field Editor will be degraded if a single line of XML is larger than 50,000 bytes. However, this problem will not arise if the XML is on multiple lines (within reason).

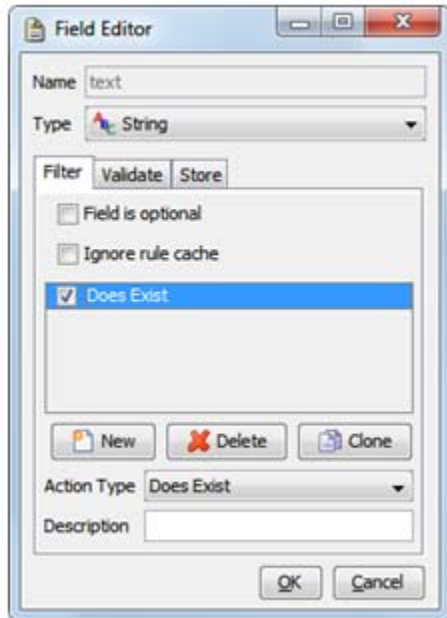


---

The following sections describe how to use the tabs on the Field Editor dialog box.

### Filter Tab

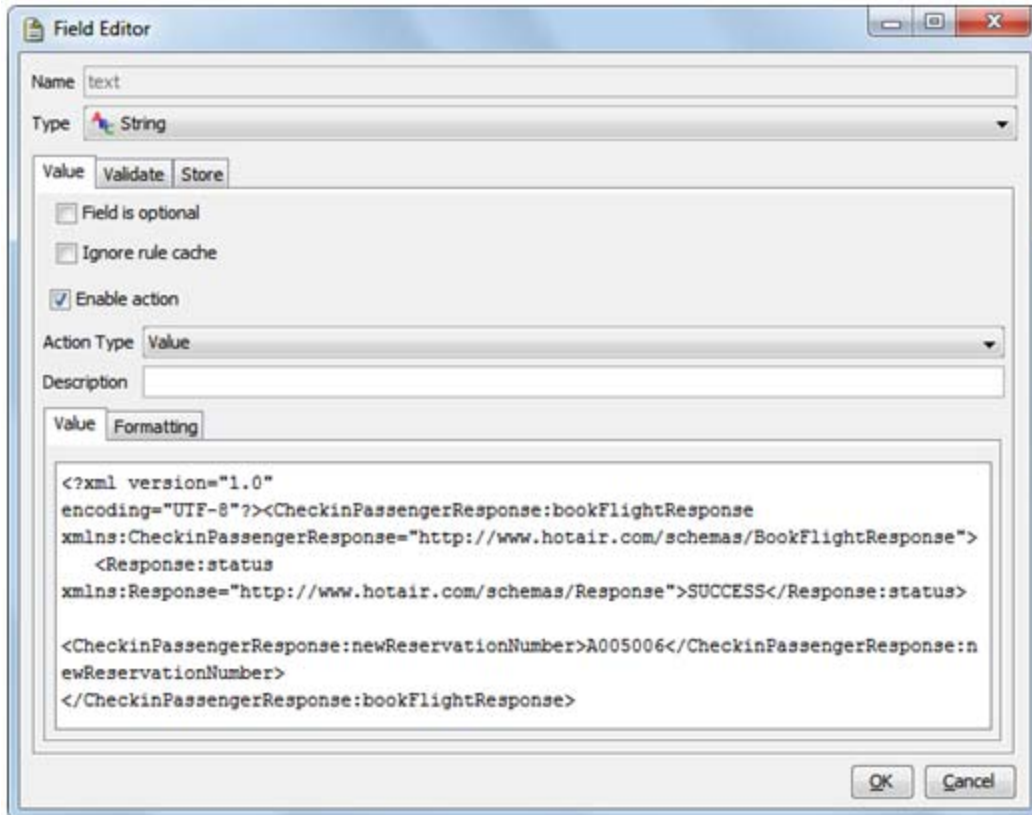
The **Filter** tab enables you to restrict which messages a subscriber will process based on the content of the selected field.



---

## Value Tab

The **Value** tab enables you to specify what value the field should take (for example, when part of a published message).



Under the **Formatting** tab (on the **Value** tab), various types of formatting can be applied to field values to generate random values at runtime.

Field Editor

Name

text

Type

String

Value

Validate

Store

☐ Field is optional
   
☐ Ignore rule cache
   
☒ Enable action

Action Type

Value

Description

Value

Formatting

☐ Enable
   

Category:

Sample

Date Time

String

Number

Currency

Custom

<?xml version="1.0" encoding="UTF-8"?><CheckinPassengerResponse:bookFlig...
 

Tag Value...

Configuration

Output Format:

SimpleDateFormat Summary

To learn more about Simple Date Formats please refer to the [GH Tester Reference Guide \(Appendix D\)](#)

Letter	Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96

Alignment

Length:

☒ None
 ☐ From Schema
 ☐ Custom
 

10

Justification:

☒ Left
 ☐ Right
 ☐ Centered

Character:

Left

Right

☒ Trim if too long?

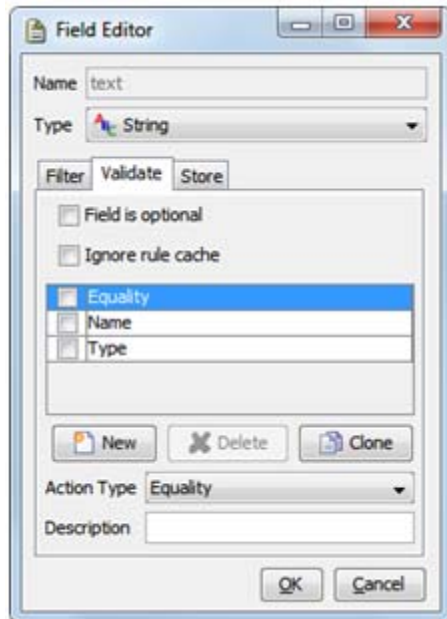
OK

Cancel

---

## Validate Tab

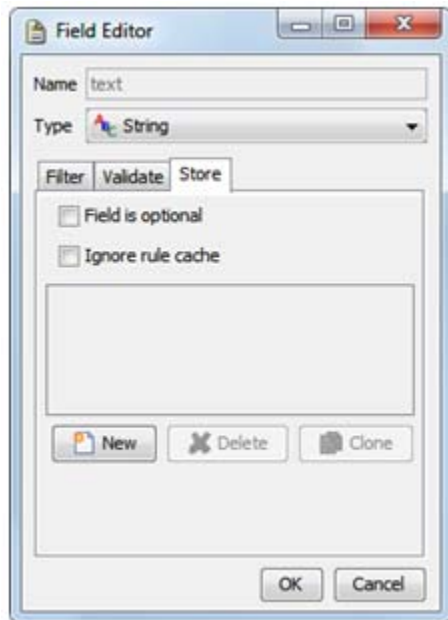
The **Validate** tab enables you to define how to compare or verify the contents of a field that requires validation (for example, in the context of subscription).



---

## Store Tab

The **Store** tab enables you to store the contents of the field in a tag (for example, when receiving a message, or to record the value of a dynamic field in a published message).



## Action-Types

There is only one way at a time to populate a value but there are many different ways to validate or store a value. Therefore, the **Validate** and **Store** tabs can accommodate one or more action-types.

Action-Type	Message-Types	Field Editor Tabs	Description
Does Exist	Incoming	Filter	Accept or ignore incoming messages depending on whether the selected field exists in each message.
Is Null	Incoming, Outgoing	Filter, Validate	Verify whether the selected field is null.
Not Null	Incoming, Outgoing	Filter, Validate	Verify whether the selected field contains a value.

---

Action-Type	Message-Types	Field Editor Tabs	Description
Name	Incoming	Filter	Accept or ignore incoming messages depending on the name of the selected field in each message.
Type	Incoming	Filter	Accept or ignore incoming messages depending on the type of the selected field in each message.
Assert using Function	Incoming, Outgoing	Filter, Validate	Use a Rational Integration Tester function (that returns a Boolean type) to filter/validate the selected field.
Validate Element Children	Incoming	Validate	Validate the children of the selected field.
Validate Using Tag	Incoming	Validate	Validate a received value against a message or sub-message that has been previously tagged.
Value	Outgoing	Value	Validate the contents of the selected of each outgoing message against a specific manually entered value.
File	Outgoing	Value	Populate the contents of the selected field with the contents of a selected file.  You must also select the file.

---

---

<b>Action-Type</b>	<b>Message-Types</b>	<b>Field Editor Tabs</b>	<b>Description</b>
Function	Outgoing	Value	Execute a Rational Integration Tester function (including custom functions).  The result of the function is used as the field value.
Decrement	Outgoing	Value	Decrease the selected field's value by a specified amount each time that the outgoing message is sent.  This involves setting an initial value and a step value.
Increment	Outgoing	Value	Increase the selected field's value by a specified amount each time that the outgoing message is sent.  This involves setting an initial value and a step value.

---

---

Action-Type	Message-Types	Field Editor Tabs	Description
List	Outgoing	Value	<p>Rational Integration Tester treats each entry in the specified input as a separate value.</p> <p>Values can be separated using any available delimiters (new line, comma, tab, or full stop).</p> <p>The first time that the selected outgoing message is sent, the selected field will be set to the first value.</p> <p>For each subsequent iteration, the next value in the sequence will be used.</p> <p>If a message is published more times than the number of available values, the cycle will restart from the first value.</p> <p>You can also reset the action and restart from the first value.</p>
Null	Outgoing	Value	<p>Clear the contents of the selected field.</p>
Equality	Outgoing	Validate	<p>Verify whether each outgoing message contains a value in the selected field that matches the specified value.</p>

---



---

Action-Type	Message-Types	Field Editor Tabs	Description
Length	Outgoing	Validate	Verify whether each outgoing message contains a value in the selected field that falls within the specified maximum and minimum values.
Regex	Outgoing	Validate	Use the specified regular expression to validate the contents of the selected field of each outgoing message.
XPath	Outgoing	Validate	Use the specified XPath expression to validate the contents of the selected field of each outgoing message.
Schema	Outgoing	Validate	Use the specified schema to validate the contents of the selected field of each outgoing message.
XSD Type	Outgoing	Validate	Verify whether the field is of the correct type as specified by the XSD used to build the message.

---

**NOTE:** The availability of action-types depends on the transport, formatter, and field-type selected.

---

## Buttons

The following table describes the default buttons on the tabs on the Field Editor dialog box.

Button	Description
New	Add a new validation to the selected field.
Delete	Delete the selected validation from the selected field.
Clone	Duplicate the selected validation for the selected field. <b>NOTE:</b> You cannot duplicate <b>Name</b> and <b>Type</b> validations.

Depending on the action-types selected on some tabs, other buttons are also displayed.

## Check Boxes

The following table describes the default check boxes on the tabs on the Field Editor dialog box.

Check Box	Description
Field is optional	
Ignore rule cache	Ignore a specific action-type within a rule that has been set on the selected field
Does Exist	
Accept fields in any order	Rational Integration Tester will ignore the order in which message fields are received. This can be useful since messages sent on certain transports may undergo field-reordering.
Ignore missing fields in received messages	When enabled, any fields present in the expected message structure but absent in the message received will not cause invalidation.
Ignore additional fields in received messages	When enabled, any fields present in the received message but absent in the expected message structure will not cause invalidation.
Enable action	
Enable (Formatting)	

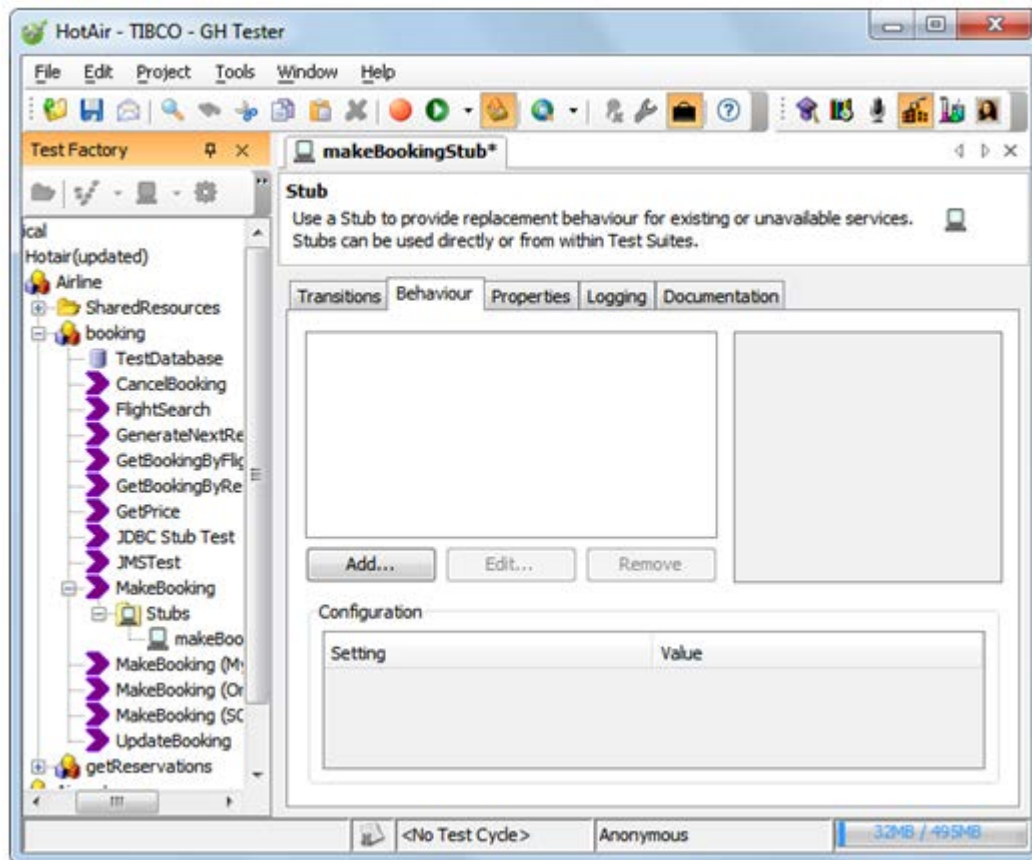
Depending on the action-types selected on some tabs, other check boxes are also displayed.

---

---

### 3.3.2 Behaviour Tab

This optional tab enables you to add reusable behavioural entities, that is, preprogrammed intelligence, to your stub.



**NOTE:** This tab is not displayed when editing a deterministic stub.

A behaviour can act as a source of events that cause a stub to execute an operation that is not a reaction to an incoming message from an external system. For example, you might want to create a stub that can publish proactively messages that are not just “responses” to a incoming message.

After you have defined one or more behaviours on the **Behaviour** tab, you can use them as “events” on the **Transitions** tab.

The upper half of the **Behaviour** tab comprises:

- A window that lists any behaviours created for the stub, and buttons for creating, modifying, and deleting behavioural entities.

- 
- A text box that displays a description for each selected behaviour.

The lower half of the tab comprises a Configuration window that displays details about any behavioural entity with custom instance names.

The following table outlines the default behaviour-types that are supplied with Rational Integration Tester.

---

Behaviour-Type	Description
Lifecycle	This enables you to execute certain actions when the currently selected stub starts up or shuts down. For example, you can set up and later clean up any resources that may be required by the stub.
Timer	This enables you to set up a timer that schedules a future callback after a specified delay so that the stub will receive an event to which it can respond. <b>NOTE:</b> Callback timer settings for a stub can be overridden for each transition created for the stub (for information about this, refer to <a href="#">Transitions Tab</a> ).

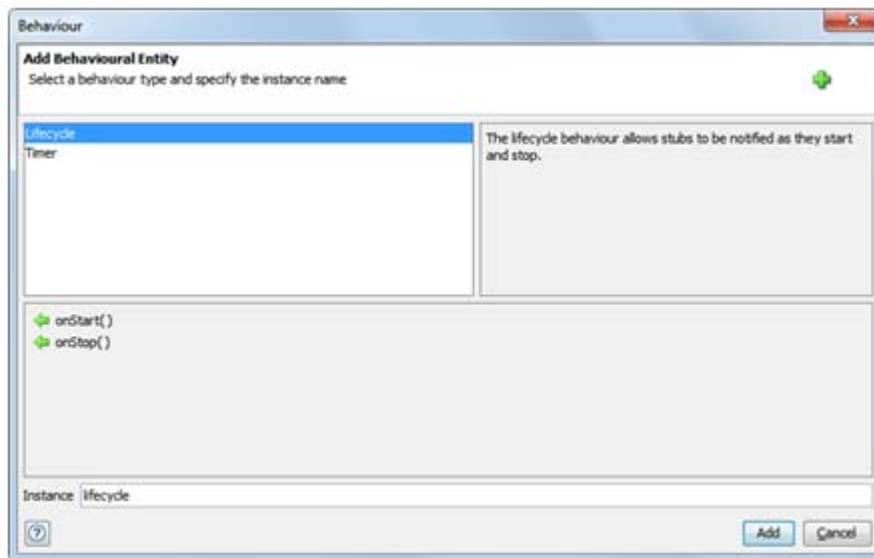
---

**NOTE:** Rational Integration Tester provides the means to create additional custom behaviours. For more information about this, refer to [Appendix A: Using the Data Model Editor](#).

---

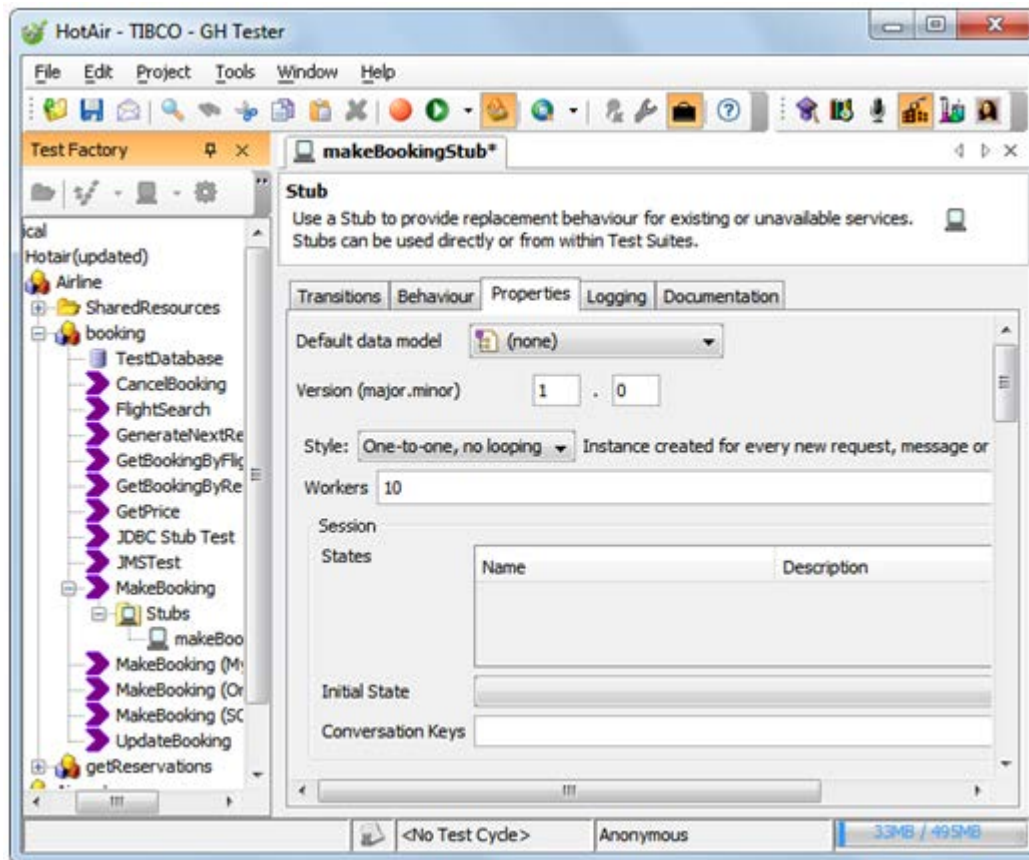
The following table describes the buttons on the upper half of the **Behaviour** tab.

Button	Description
Add	<p>Clicking this button opens the Add Behavioural Entity dialog box.</p> <p>The upper left side of the Add Behavioural Entity dialog box displays the available behaviour-types. The upper right side of the Add Behavioural Entity dialog box displays a description for each selected behaviour-type.</p> <p>The default available behaviour-types are as follows:</p> <ul style="list-style-type: none"><li>• <b>Lifecycle</b></li><li>• <b>Timer</b></li></ul> <p>The lower half of the Add Behavioural Entity dialog box displays the available behaviours for the currently selected behaviour-type and an <b>Instance</b> field, which is an optional field that enables you to create multiple instances of a behaviour.</p> <p><b>NOTE:</b> To select multiple behaviours for a behavioural entity, hold down CTRL and click each behaviour after selecting a behaviour-type on the upper left side of the Add Behavioural Entity dialog box.</p> <p>After specifying the properties of a behaviour, click <b>Add</b> to complete creating the behaviour and to close the Behaviour dialog box.</p> <p><b>NOTE:</b> In addition to being displayed on the list window on the upper left side of the <b>Behaviour</b> tab, each newly created behavioural entity is also added to the <b>Actor</b> list on the <b>Transitions</b> tab for the stub (for information about the <b>Actor</b> field, refer to refer to <a href="#">Transitions Tab</a>).</p>
Edit	<p>Clicking this button opens the Edit Behavioural Entity dialog box, which enables you to modify instance details for the selected behavioural entity.</p>
Delete	<p>Clicking this button deletes any behavioural entities selected in the list window on the tab.</p> <p><b>NOTE:</b> To select multiple behavioural entities for deletion, hold down CTRL and click each entity.</p>



### 3.3.3 Properties Tab

This tab enables you to set up states for your stub and input parameters that can determine how the stub should behave while it is executed.



The upper half of the tab comprises fields and lists that enable you to specify how the stub will execute, the number of concurrent instances that can be running at any one time, the session states that should be available to the stub, and the response times that control the performance of the stub.

The following table describes the fields and lists on the upper half of the **Properties** tab.

Field/List	Description
Default data model	This list is used to specify the default data model (if any) of the stub.
Version (major.minor)	These fields enable you to implement a version numbering system for your stubs (if necessary).

---

Field/List	Description
Style	<p>This list enables you to control how the stub will execute.</p> <p>Options are as follows:</p> <ul style="list-style-type: none"><li>• <b>One-to-one, no looping:</b> An instance is created for every new request, message, or connection.</li><li>• <b>One-to-one, looping:</b> An instance is created for each new connection, and the instance lives until the connection closes. This behaviour is applies to connection-based transports.</li><li>• <b>Many-to-one:</b> An instance is created if no instance is currently available to process the request or messages. Worker threads are not enabled for this behaviour.</li></ul>
Workers	<p>This field specifies the maximum number of concurrent instances of this stub that can be running at any one time.</p> <p>Default value: 10.</p>
States	<p>This field specifies the session states that should be available for the stub. These will be added automatically when you add states on the <b>Transitions</b> tab.</p> <p>One or more states can be entered in this field by:</p> <ul style="list-style-type: none"><li>• Clicking <b>New</b> beside the field.</li><li>• Entering details about each state in the New State dialog box.</li><li>• Clicking <b>OK</b>.</li></ul> <p><b>NOTE:</b> You can also add new states on the <b>Transitions</b> tab by selecting any row under <b>From</b> or <b>To</b> on the list table and clicking <b>Add new...</b></p> <p>When a message case is used within the stub, the state of the stub can be checked before any actions in the case are executed, and a new state can be optionally set after the actions in the case have complete executing.</p> <p>The session state is also stored in a system tag named <code>SESSION/STATE</code>. If desired, the session state could be set by using the <code>setTag</code> function.</p>
Initial State	<p>This field specifies the stub's initial state.</p>

---



---

---

Field/List	Description
Conversation Keys	<p>This field specifies the conversation (session) keys that can be used to identify a particular conversation.</p> <p>For example, if you are building a stub that can track user sessions, you might want to have a conversation key called <code>SessionID</code>.</p> <p>One or more keys (separated by a semicolon) can be defined. These keys are mapped to a system tag named <code>SESSION/KEY/&lt;Key Name&gt;</code>.</p> <p>When an incoming message is received, key values should be stored in these tags, which are then used to identify the conversation to use.</p>
Delay Distribution	<p>This list controls the stub's performance, enabling you to create realistic system conditions in a single stub (that is, without having to create multiple stub instances).</p> <p>Delay pattern options are as follows:</p> <ul style="list-style-type: none"><li>• Fixed</li><li>• Uniform</li><li>• Gaussian</li></ul>
Minimum Delay (ms)	This field specifies the minimum delay time (in milliseconds) for the delay-type selected in the <b>Delay Distribution</b> list.
Maximum Delay (ms)	This field specifies the maximum delay time (in milliseconds) for the delay-type selected in the <b>Delay Distribution</b> list.

---

The lower half of the tab enables you to define the input tags that should be available to the stub at run time if the stub is part of a test suite.

**NOTE:** Output tags are used to pass information between tests. That is, the output tags of one test can be connected to the input tags of another test. However, output tags do not apply to stubs because you cannot define any destinations for the tags. Therefore, you should ignore the **Output** section of the lower half of the tab.

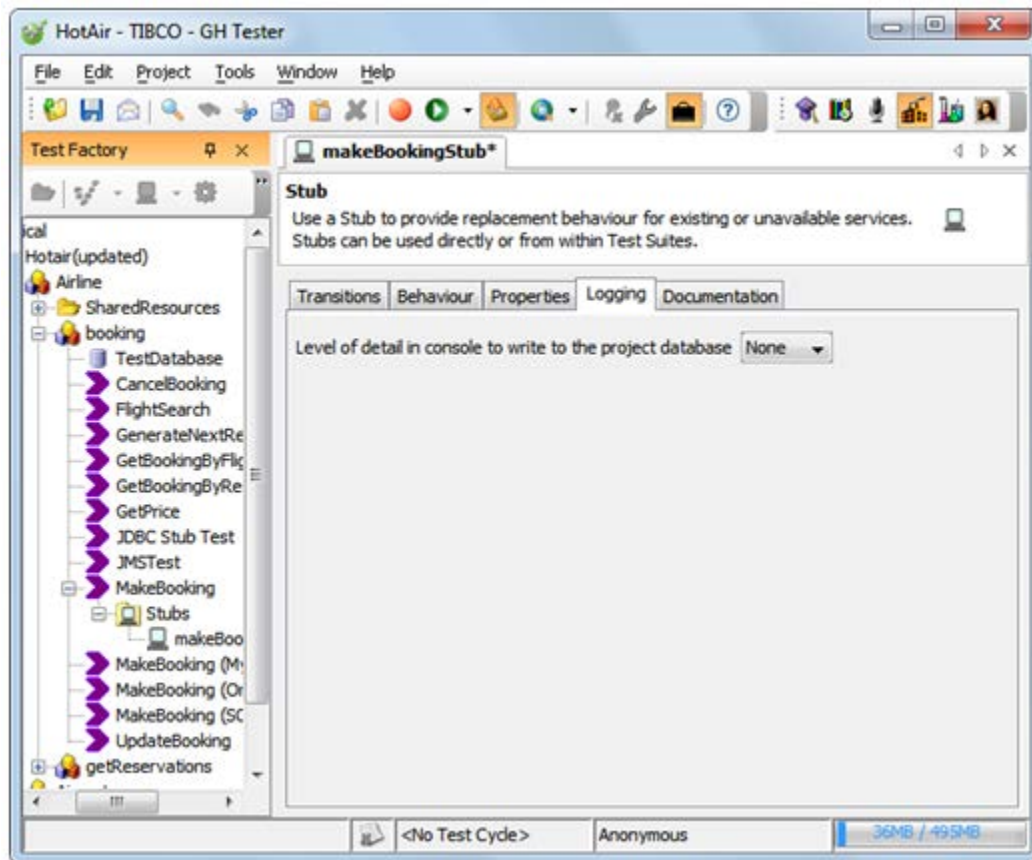
In addition, when a tag is being created, the interface definition can be set by selecting the **Expose as input** check box on the Create Tag dialog box.

For more information about creating and managing tags, refer to *IBM Rational Integration Tester Reference Guide*.

---

### 3.3.4 Logging Tab

This tab enables you to define how much information is recorded while your stub is being executed.



Any information recorded will be logged in Rational Integration Tester's project results database and on the Test Factory perspective's Console window.

The following table describes the list options on the **Logging** tab.

---

List Option	Description
None	No information is logged.
Normal	Standard information is logged.
Debug	Verbose information is logged.

---

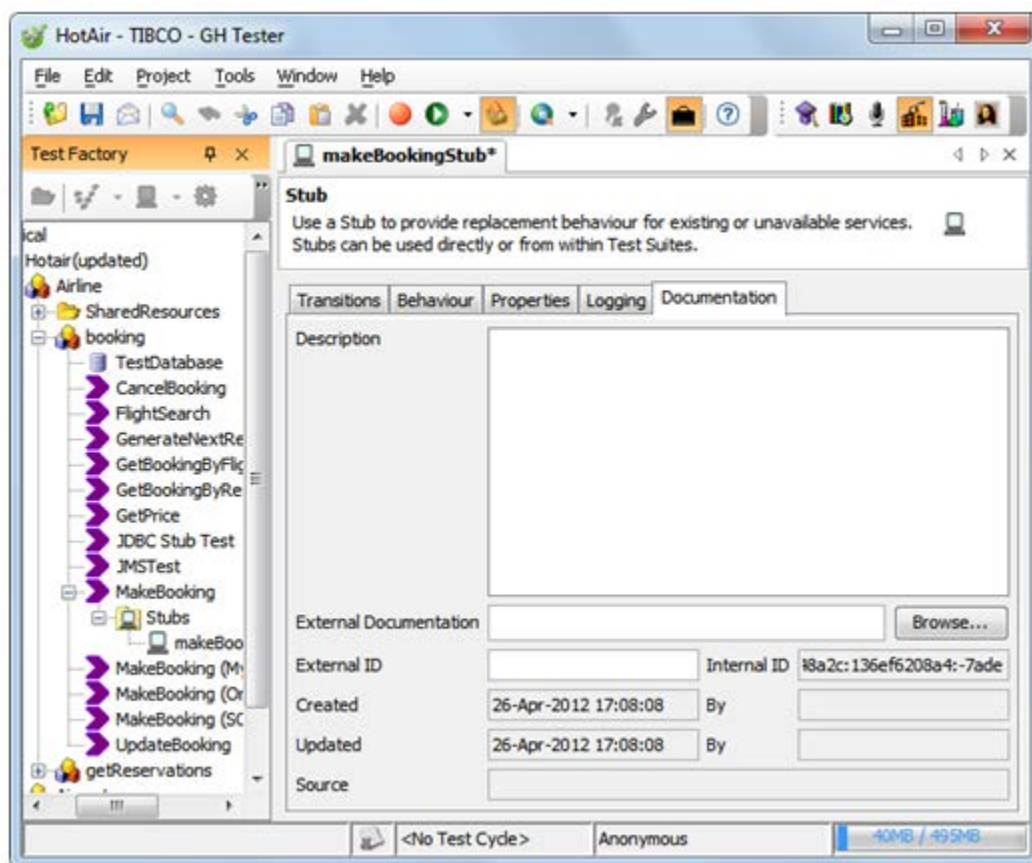
**NOTE:** The default list option displayed on the **Logging** tab is determined by the setting of the **Stub default log level** list on the **Console** page of

---

the Preferences dialog box, which is opened by clicking **Project > Preferences** or **Window > Preferences** on the menu bar. Therefore, selecting (for the currently selected stub) a different log level on the **Logging** tab will overwrite the default log level specified in the Preferences dialog box but only for that stub.

### 3.3.5 Documentation Tab

This tab enables you to enter extra information about your stub and some of this extra information will be displayed in Rational Test Control Panel's Start Stub dialog box (for information about this dialog box, refer to [Starting Stubs](#)).



Although you do not have to enter such information, it may help future users of your stub and it may also help you to identify in Rational Test Control Panel which stub you want to start if you have created several stubs for a particular operation.

For more information about using this tab, which is a standard tab on many dialog boxes in Rational Integration Tester, refer to *IBM Rational Integration Tester Reference Guide*.

# Creating & Modifying Database Stubs

## **Contents**

### **Introduction**

### **Before Creating Database Stubs**

### **Creating Database Stubs**

### **Modifying Database Stubs**

This chapter describes how to create and maintain database stubs.

For information about recording SQL events but not creating database stubs, refer to *IBM Rational Integration Tester Reference Guide*.

---

## 4.1 Introduction

The following sections provide an introduction to database stubs.

### 4.1.1 Purpose

If you want to create a test or suite of tests for an application that uses a database, you will need to be able to run the test (suite) in a repeatable fashion against a known set of database contents. Therefore, you will need to stub the database to obtain repeatable conditions.

### 4.1.2 Benefits

Database stubs enable you to execute tests against some parts of a system under test without affecting a live database.

### 4.1.3 Key Concepts

The following sections describe key database stubbing concepts in Rational Test Virtualization Server.

#### 4.1.3.1 Persistence

Database stubs created in Rational Test Virtualization Server can be non-persistent or persistent:

- A non-persistent database stub will start from the same known state each time it is run. That is, any changes made to the contents of a database stub during its use will be lost when the stub is stopped. This enables tests to run against a known starting state.
- A persistent database stub will remember its state when it is stopped, overwriting the previous saved state. When it is restarted, the stub will have the same data that it had when it was last stopped.

It is possible to change a stub from being persistent to being non-persistent. This is useful if you want a stub to be persistent while you design, build, edit, and refine it for a test case; and then non-persistent, and thus in a known start state, each time that it is used.

The following sections describe how to use Rational Test Virtualization Server to create, run, and publish database stubs. (For information about recording SQL but not creating or using database stubs, refer to *IBM Rational Integration Tester Reference Guide*.)

---

#### 4.1.3.2 Rational Integration Tester JDBC Proxy Modes

The Rational Integration Tester JDBC proxy enables Rational Integration Tester and Rational Test Virtualization Server to:

- Record SQL executed against databases from applications that use JDBC.
- Create and edit database stubs (Rational Test Virtualization Server only).

Database stubs contain subsets of data from a “live” (production) database. The contents of the stubs are built by analyzing an application’s use of SQL against the live database.

- Start a database stub (Rational Test Virtualization Server only).

Starting a stub loads the stub data into a simulation database and transparently redirects the application to that simulation database.

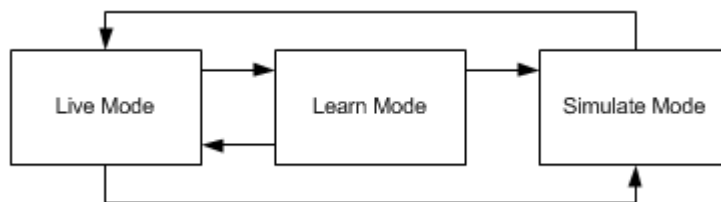
Therefore, users can test JDBC applications in a more deterministic manner.

When creating database stubs, the following modes of the Rational Integration Tester JDBC proxy are relevant:

- Live
- Learn
- Simulate

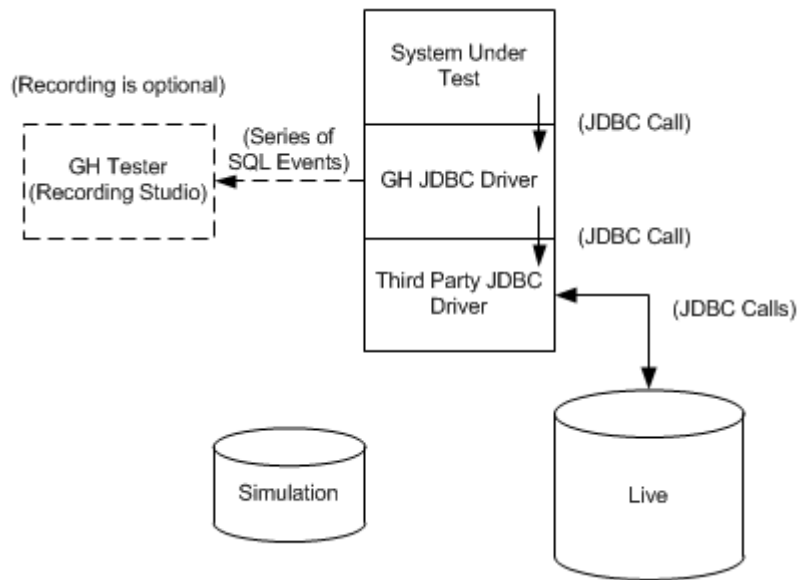
The purpose of learn mode is to fill the simulation database, which facilitates creation of database stubs. Creating a stub then starting it moves learn mode to simulate mode, and stopping the stub moves simulate mode to live mode. When in live mode, it is possible to enter learn mode again.

The following diagram illustrates the lifecycle for these modes.



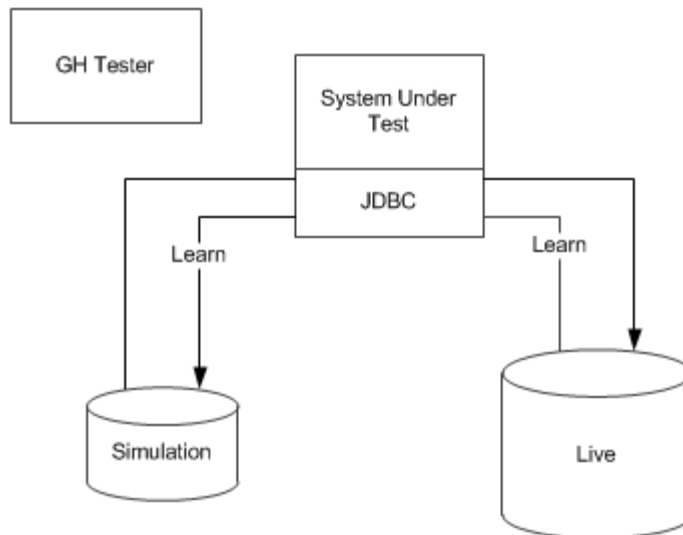
---

The following diagram illustrates live mode.

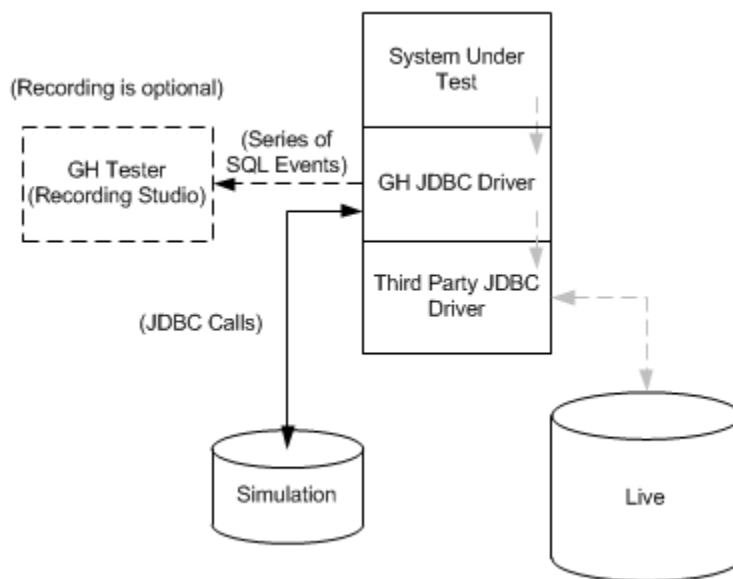


---

The following diagram illustrates learn mode.



The following diagram illustrates simulate mode.



The following sections describe how to set up, create, start, stop, and modify database stubs.



---

## 4.2 Before Creating Database Stubs

Before creating any database stubs, you must complete any required software installation tasks and you must set up the schema that will be used for stubbing a physical database.

The following sections describe how to complete these prerequisites.

### 4.2.1 Preparation and Planning

The remainder of this chapter assumes that you have installed and configured the required software.

For information about installing and configuring the required software, and determining which live databases are to be stubbed and whether there are any database schema or data source requirements, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.

### 4.2.2 Setting Up the Schema Used for Stubbing a Physical Database

Before you can create database stubs, you must specify in Rational Integration Tester which database schema is to be used for the simulation database.

This schema will be used while Rational Integration Tester is “learning” the stubs and it will also be used when a stub is started, that is, the database schema that will be used to store the contents of a database stub.

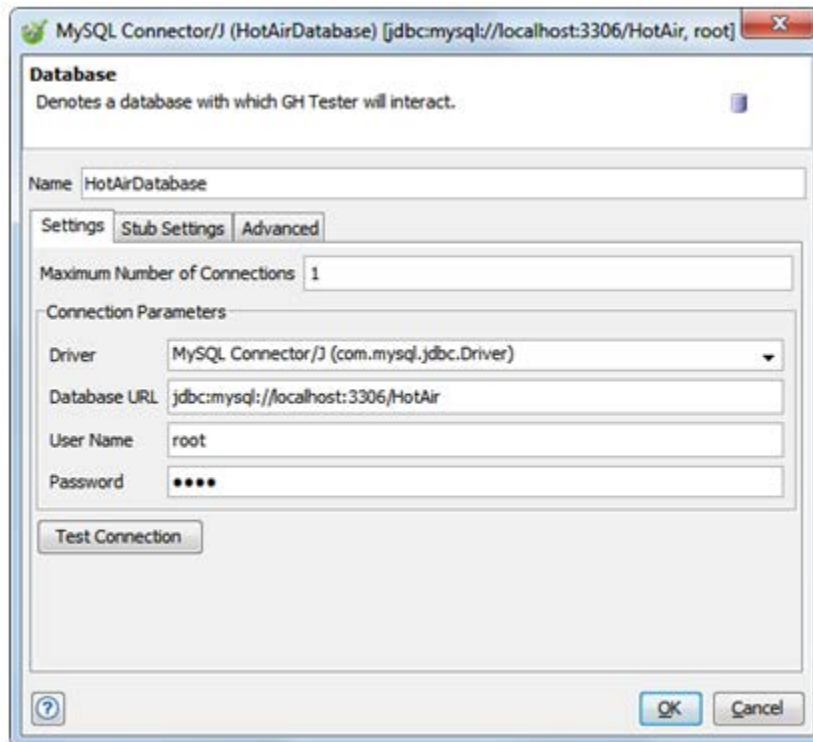
**NOTE:** If you want to run more than database stub simultaneously, you will need a database stub schema for each database stub. For example, if you want to run five database stubs simultaneously, you will need five database stub schemas. However, each schema must match the corresponding database. For example, if you want to stub an Oracle database, you will need an Oracle schema. Alternatively, if you want to stub a Microsoft SQL Server database, you will need a Microsoft SQL Server schema, and so on.

---

To set up the schema that will be used for stubbing a specific physical database:

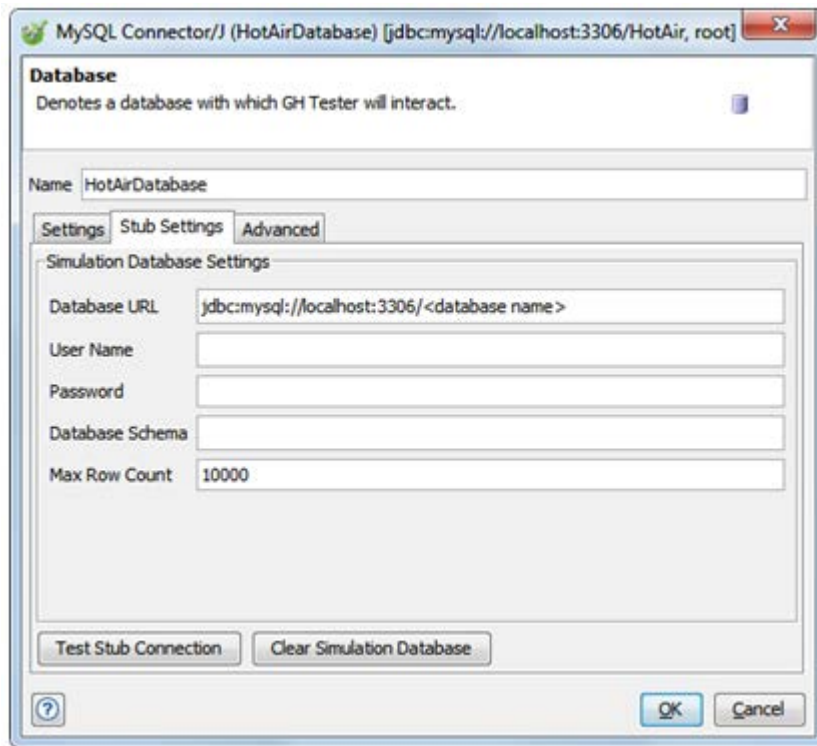
1. In Rational Integration Tester's Architecture School perspective's Physical View, double-click the physical database that you want to stub.

The Database (Properties) dialog box is displayed.

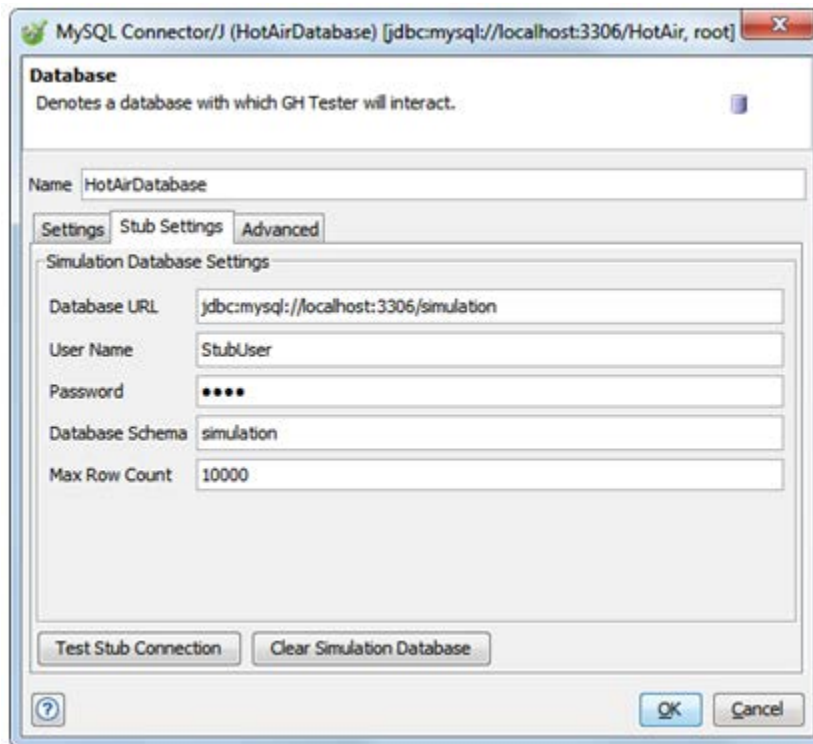


2. Click the **Stub Settings** tab.

The **Stub Settings** tab is displayed.



3. In the **Database URL** field, enter the URL of the server that will host the database stub schema.
4. In the **User Name** field, enter the name of the relevant database user.
5. In the **Password** field, enter the database user's password.
6. In the **Database Schema** field, enter the name of the database stub schema.



**NOTE:** The schema must not be the schema used by the live database. In addition, the user name, password, and schema name should be provided by a database administrator (DBA). The database user must also have sufficient privileges to be able to create and delete tables in the simulation database.

7. Click **Test Stub Connection** to verify Rational Integration Tester's connection to the database stub.
8. Click **OK**.

You are now ready to create database stubs.

---

## 4.3 Creating Database Stubs

Rational Test Virtualization Server provides users with two methods of creating a database stub:

- Use Rational Integration Tester's Recording Studio.
- Change the Rational Integration Tester JDBC proxy's modes manually.

The following sections describe how to use each of these methods.

### 4.3.1 Recording Studio Method

Under this method, Rational Integration Tester's Recording Studio perspective is used to create a database stub while SQL events are being recorded from a database.

To create a database stub, you must record SQL statements from a live application to populate the stub with data from the live database. Each SQL `SELECT` statement that is recorded will be analyzed by Rational Integration Tester and the corresponding results from the live system will be copied into the database stub that is being "learned" during recording. Rational Integration Tester attempts to copy the data matching the SQL `WHERE` clause.

For example, if the recording contains only the reading of one customer, the database stub will contain only data about that customer. Alternatively, if the recording contains the reading of, say, all customers with first name "John", the database stub will contain the same data.

**NOTE:** If there is no primary key on a database table that is being "learned", the simulation database is will probably contain duplicate records when learning is complete. However, Rational Integration Tester will display error messages if it detects any table that do not have any primary keys.

To create a database stub from recorded SQL events:

1. In Rational Integration Tester's Recording Studio perspective, select (for recording) the database that you want to stub.

**NOTE:** The database selected must be the database that was specified in [Before Creating Database Stubs](#).

2. Click the **Record** button (●) on the Events View toolbar.

The Create Stubbed Database Whilst Recording? dialog box is displayed.



3. Click the **Yes, record the SQL and create a virtual database** option button.
4. Click the **Create a database stub and start it** option button or the **Create a database stub but don't start it** option button.
5. In the **Stub name** field, enter a name for the stub.



- 
6. Click **Start Recording**.
  7. Run a test case or a test case suite against the live system to populate the stub with sufficient data to make it useful.

You will be able to add more data to the stub later. During recording, Recording Studio will display information relating to stub creation, such as messages about database table creation. However, the stub will not be created **until** you stop recording.

8. On the Create Stubbed Database Whilst Recording? dialog box, click **Stop Recording**.

If you clicked the **Create a database stub and start it** option button or the **Create a database stub but don't start it** option button, the stub is now created.

If you clicked the **Keep learning** option button, you will have to execute another (or perhaps many more) stub recording session(s) to generate sufficient SQL to create the stub.

In Rational Integration Tester's Test Factory perspective, the newly created database stub is displayed under the relevant logical resource on the component tree.

You can now start, stop, or modify the new database stub. (These operations are described elsewhere in this chapter.)

### 4.3.2 Proxy Mode Change Method

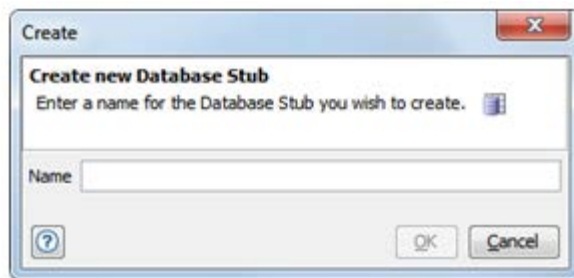
Under this method, users manage the live/learn/simulate lifecycle of the driver manually. (For information about this lifecycle, refer to [Rational Integration Tester JDBC Proxy Modes](#).)

To create and use a database stub by changing the Rational Integration Tester JDBC proxy's modes manually:

1. In Rational Integration Tester's Test Factory perspective, right-click the database that you want to stub and click **New > Stubs > Database Stub** on the shortcut menus.

**NOTE:** The database selected must be the database that was specified in [Before Creating Database Stubs](#).

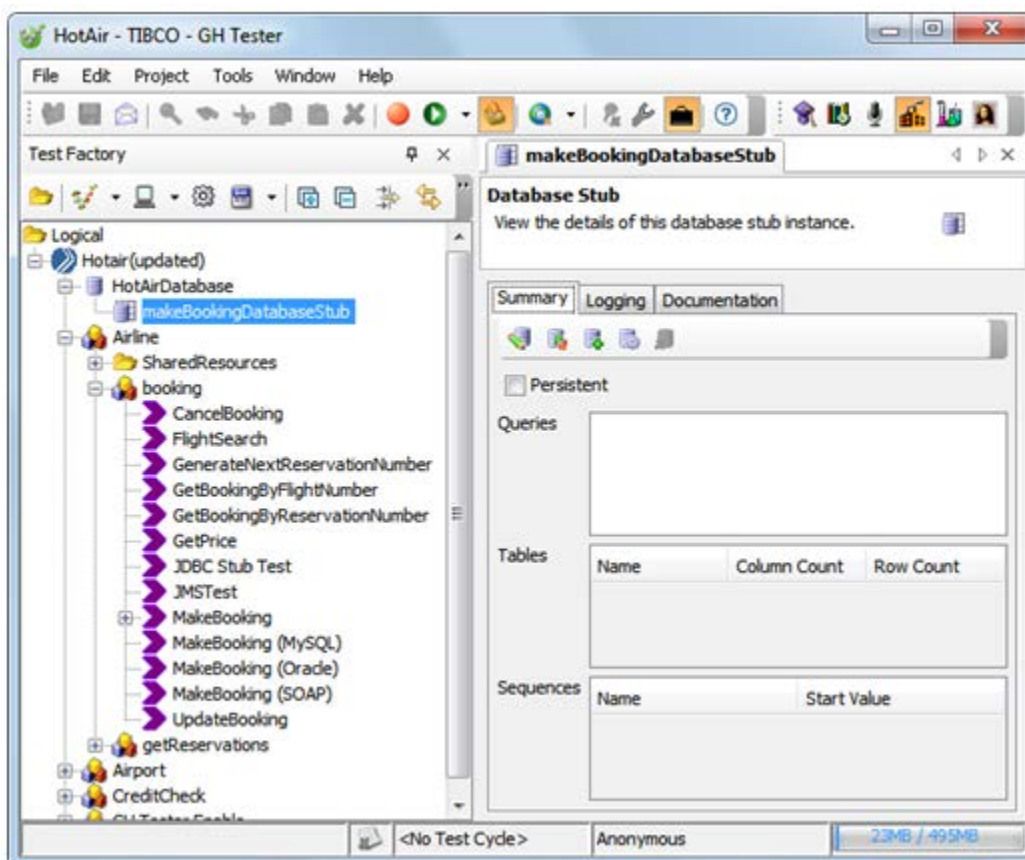
The Create a New Database Stub dialog box is displayed.



2. In the **Name** field, enter a name for the database stub.
3. Click **OK**.

The stub is displayed under the relevant logical resource on the Test Factory perspective's component tree.

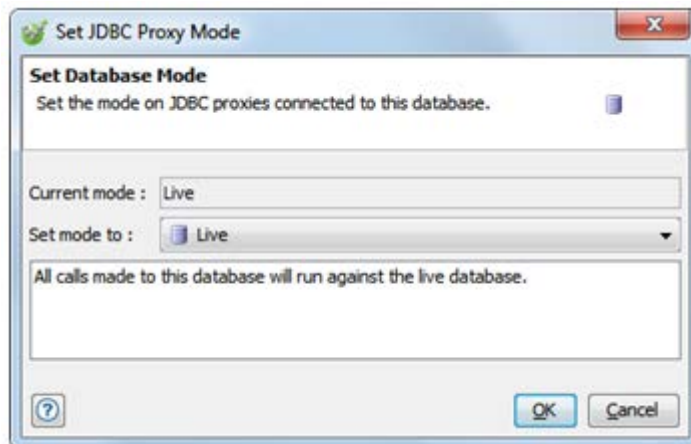
Selecting the empty stub opens it in the Database Stub Editor.



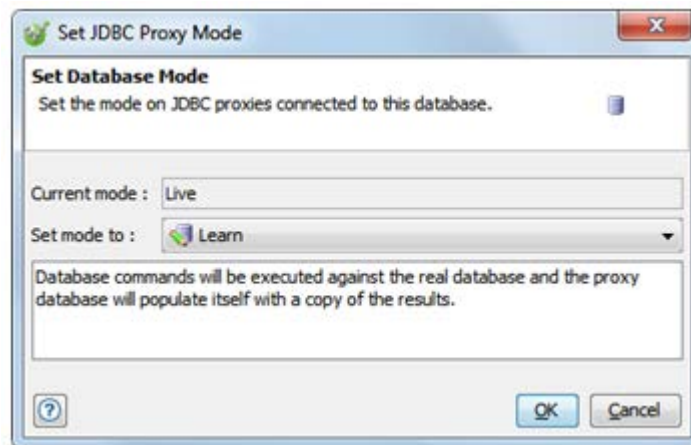


- 
4. In Rational Integration Tester's Architecture School perspective's Logical View, right-click the logical database and click **Set Proxy Mode** on the shortcut menu.

The Set JDBC Proxy Mode dialog box is displayed.



5. In the **Set mode to** list, click **Learn**.



6. Click **OK**.
7. Populate the simulation database with live database data by using any of the following methods:
  - Run any test cases created for the system under test.
  - Use the system under test.
  - Wait for the system under test to populate the live database with query results.

- 
8. After a period of time has elapsed (the period length will depend on the method chosen to populate the simulation database), open the Architecture School perspective's Logical View.

9. Right-click the database that is being stubbed and click **Set Proxy Mode** on the shortcut menu.

The Set JDBC Proxy Mode dialog box is displayed.

10. In the **Set mode to** list, click **Live**.

Rational Integration Tester will stop "learning" SQL and a stub has been created.

11. Click **OK**.

In Rational Integration Tester's Test Factory perspective, the newly created database stub is displayed under the relevant logical resource on the component tree.

You can now start, stop, or modify the new database stub.

The following sections describe these operations.

---

## 4.4 Modifying Database Stubs

When you open a database stub in Rational Integration Tester's Test Factory perspective, a Database Stub Editor screen is displayed. The following table summarizes the tabs on this screen.

Tab	Functionality
Summary	<p>This tab contains the following:</p> <ul style="list-style-type: none"><li>• An editing options toolbar, which is context sensitive (for information about this, refer to <a href="#">Editing Options Toolbar</a>).</li><li>• A <b>Persistent</b> check box for specifying whether a database stub is persistent or non-persistent (for information about this, refer to <a href="#">Persistence</a>).</li><li>• A list of the SQL queries that were recorded during learn mode.</li><li>• The tables contained in the stub. For each table, the table name and the row and column counts are displayed.</li><li>• The sequences contained in the stub. For each sequence, the name and start value are displayed.</li></ul>
Logging	<p>This tab enables you to define how much information is recorded while your stub is being executed.</p> <p>Any information recorded will be logged in Rational Integration Tester's project results database and on the Test Factory perspective's Console window.</p> <p>Options are as follows:</p> <ul style="list-style-type: none"><li>• <b>None</b> (no information is logged).</li><li>• <b>Normal</b> (standard information is logged).</li><li>• <b>Debug</b> (verbose information is logged)</li></ul>
Documentation	<p>This tab enables you to enter extra information about your stub and some of this extra information will be displayed in Rational Test Control Panel's Start Stub dialog box (for information about this dialog box, refer to <a href="#">Starting Stubs</a>).</p> <p>Although you do not have to enter such information, it may help future users of your stub and it may also help you to identify in Rational Test Control Panel which stub you want to start if you have created several for the same database.</p> <p>For more information about using this tab, which is a standard tab on many dialog boxes in Rational Integration Tester, refer to <i>IBM Rational Integration Tester Reference Guide</i>.</p>






---

The following sections describe how to use **Summary** tab.

---

### 4.4.1 Editing Options Toolbar

The following table describes how to use the toolbar on the **Summary** tab.

To...	Do this...
Use a worksheet to edit the currently selected database stub	Click  . The Edit Database Stub wizard is displayed. For information about using this wizard, refer to <a href="#">Edit Action</a> .
Import additional data from a live database to the currently selected database stub	Click  . The Learn wizard is displayed. For information about using this wizard, refer to <a href="#">Learn Action</a> .
Add additional queries, tables, or sequences to the currently selected database stub	Click  . The Add wizard is displayed. For information about using this wizard, refer to <a href="#">Add Action</a> .
Use an external application, such as a database management system tool, to edit the currently selected database stub	Click  . The External Edit wizard is displayed. For information about using this wizard, refer to <a href="#">Edit Using an External Tool Action</a> .
Delete one or more queries, tables, or sequences from the currently database stub	Click  . The Delete wizard is displayed. For information about using this wizard, refer to <a href="#">Delete Action</a> .

---

---

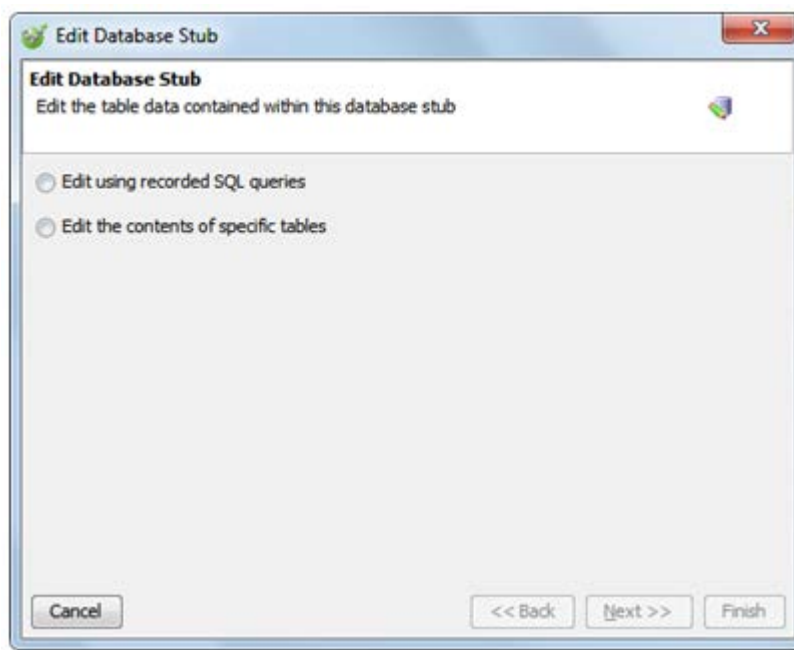
## 4.4.2 Actions

The following sections describe how to edit a database stub.

### 4.4.2.1 Edit Action

The Edit action enables you to edit the contents of the currently selected database stub.

Selecting the Edit action from the toolbar on the **Summary** tab opens the first screen of the Edit Database Stub wizard, which enables you to choose the set of queries or tables that you want to use to edit the currently selected database stub.



**NOTE:** Right-clicking the **Queries**, **Tables**, or **Sequences** window (as appropriate) on the Database Stub Editor screen and clicking **Edit** on the shortcut menu enables you to bypass the first screen of the wizard.

After you have chosen the set of queries that you want to edit, a spreadsheet containing those queries is displayed.

You can edit any value that you select in the spreadsheet. You can also add columns to the tables by entering a new column name in the header cell along with the database-type to be used for the column, and then entering values for each row. (The type name should be in parentheses. For example: `NEW_COL ( VARCHAR ( 20 ) )`.)

---

**NOTE:** You cannot delete columns from any tables.

The wizard remains open until you click **Next** or **Finish**. After you have closed the wizard, a summary page is displayed.

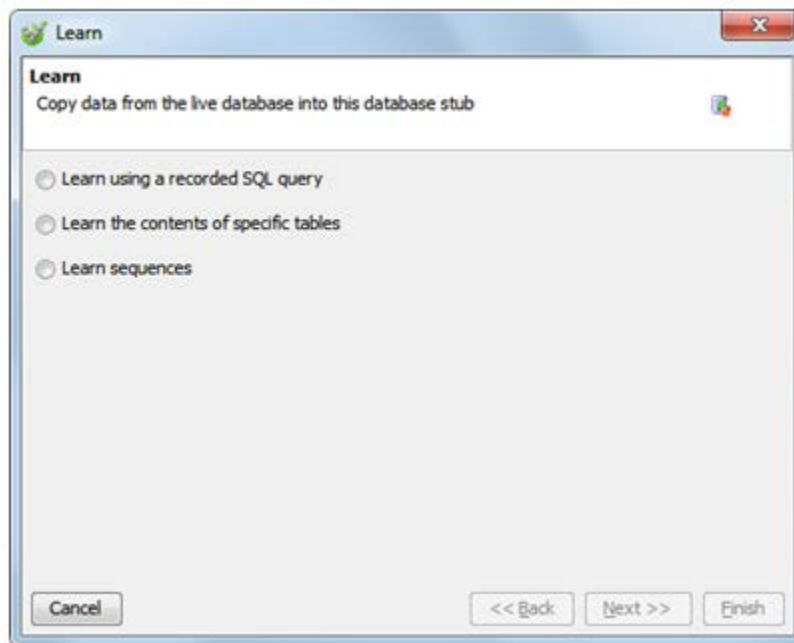
To edit a sequence's start value, click the value on the Database Stub Editor screen and then modify it.

**NOTE:** If you want to edit any selected queries or tables quickly, double-clicking them opens a spreadsheet editor with the selected subset of data. To complete the editing of the data, close the spreadsheet. However, if you choose this editing method, you will not be able to customize any selected queries.

#### 4.4.2.2 Learn Action

The Learn action enables you to import additional data from a live database to the currently selected stub.

Selecting the Learn action from the toolbar on the **Summary** tab opens the first screen of the Learn wizard.



---

The following table describes the options on the first screen of the wizard.

Option Button	Description
Learn using a recorded SQL query	This enables you to select one or more of the recorded queries and use them to copy data from the live database into this stub. You can edit the queries to refine/filter the data that will be copied, and you can delete matching data from the stub before copying the new data across.
Learn the contents of specific tables	This enables you to select one or more tables and to build simple <code>SELECT * FROM TABLE</code> queries for them. You can refine/filter the queries and delete matching data before copying the new data across.
Learn sequences	This sets the start value of one or more sequences to match the values in the live database.

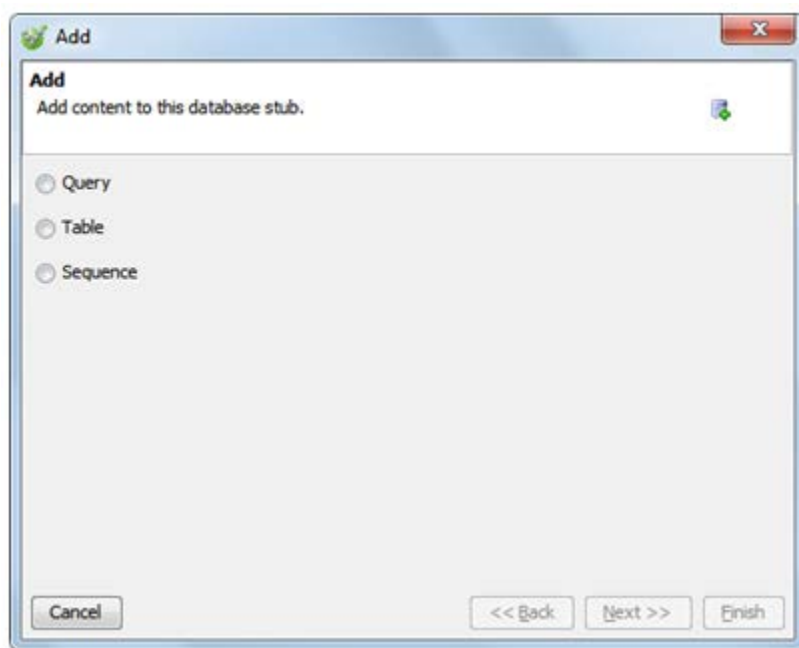
**NOTE:** Right-clicking the **Queries**, **Tables**, or **Sequences** window (as appropriate) on the Database Stub Editor screen and clicking **Learn** on the shortcut menu enables you to bypass the first screen of the wizard.

---

#### 4.4.2.3 Add Action

The Add action enables you to add additional queries, tables, or sequences to the currently selected database stub.

Selecting this action from the toolbar on the **Summary** tab opens the first screen of the Add wizard.



The following table describes the options on the first screen of the wizard.

---

Option Button	Description
Query	<p>This prompts you for an SQL query.</p> <p>After quitting the wizard, the query will be added to the list of recorded queries.</p> <p>In addition, if the query is a <code>SELECT</code>, you have the option of using the query to learn from the live database while you add (to?) it.</p> <p>If you select this option, you also have the option to delete matching data from the stub before learning from the live database.</p>

---



---

---

Option Button	Description
Table	<p>This prompts you with the list of tables in the live database that are not currently in the currently selected stub.</p> <p>If you select one of those tables, you have the option of copying rows from the live database as well as creating the new table.</p> <p>Alternatively, you may enter a new table name. If you enter a new table name, a spreadsheet is displayed.</p> <p>You can add columns to the spreadsheet by filling in column names and types, and you can also enter row data.</p>
Sequence	<p>This prompts you with the list of sequences in the live database.</p> <p>If you select one of those sequences, the start value field will be populated with the current value from the live database. (The start value may then be edited if you choose.)</p> <p>If you do not want to add a “real” sequence into the database stub, you can enter a new sequence name, and choose a start value.</p>

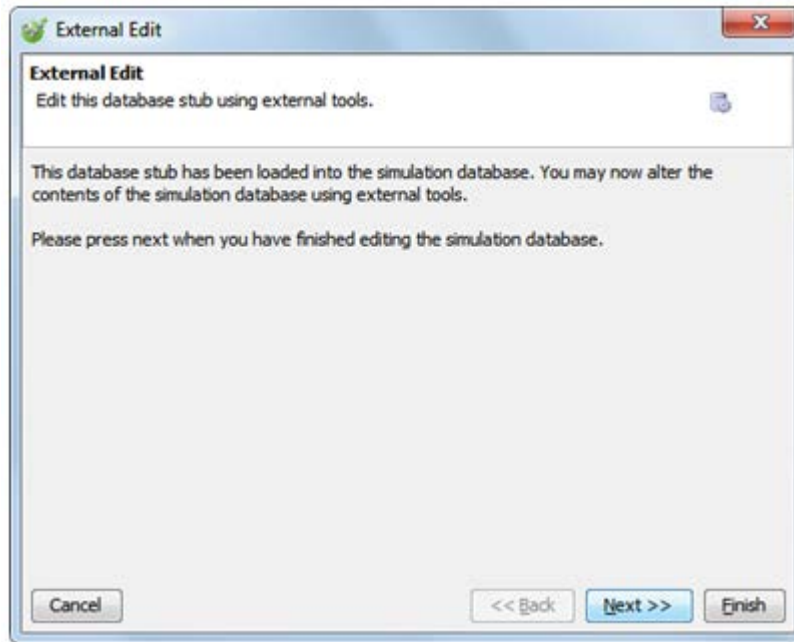
---

**NOTE:** Right-clicking the **Queries, Tables, or Sequences** window (as appropriate) on the Database Stub Editor screen and clicking **Add** on the shortcut menu enables you to bypass the first screen of the wizard.

---

#### 4.4.2.4 Edit Using an External Tool Action

The Edit Using an External Tool action enables you to use any database management systems tools (to which you have access) to edit the contents of the currently selected database stub.



Selecting this action loads the selected stub into the simulation database and the system waits for you confirm that you have completed your edits.

After you have confirmed completion of your edits, the modified stub is loaded from the simulation database.

#### 4.4.2.5 Delete Action

The Delete action enables you to delete one or more queries, tables, or sequences from the currently database stub. However, your deletions will not be executed **until** you save your changes to the stub.

# Publishing & Running Stubs

## **Contents**

### **Publishing Stubs**

This chapter describes how to publish and run stubs.

### **Running Stubs**

For a brief overview of Rational Test Control Panel, refer to [Using Rational Test Control Panel](#).

---

## 5.1 Publishing Stubs

Stubs that have been built inside Rational Integration Tester can be published to a Rational Test Control Panel instance, which facilitates deployment and control of stubs without requiring all users of those stubs to have access to Rational Integration Tester.

To publish a stub:

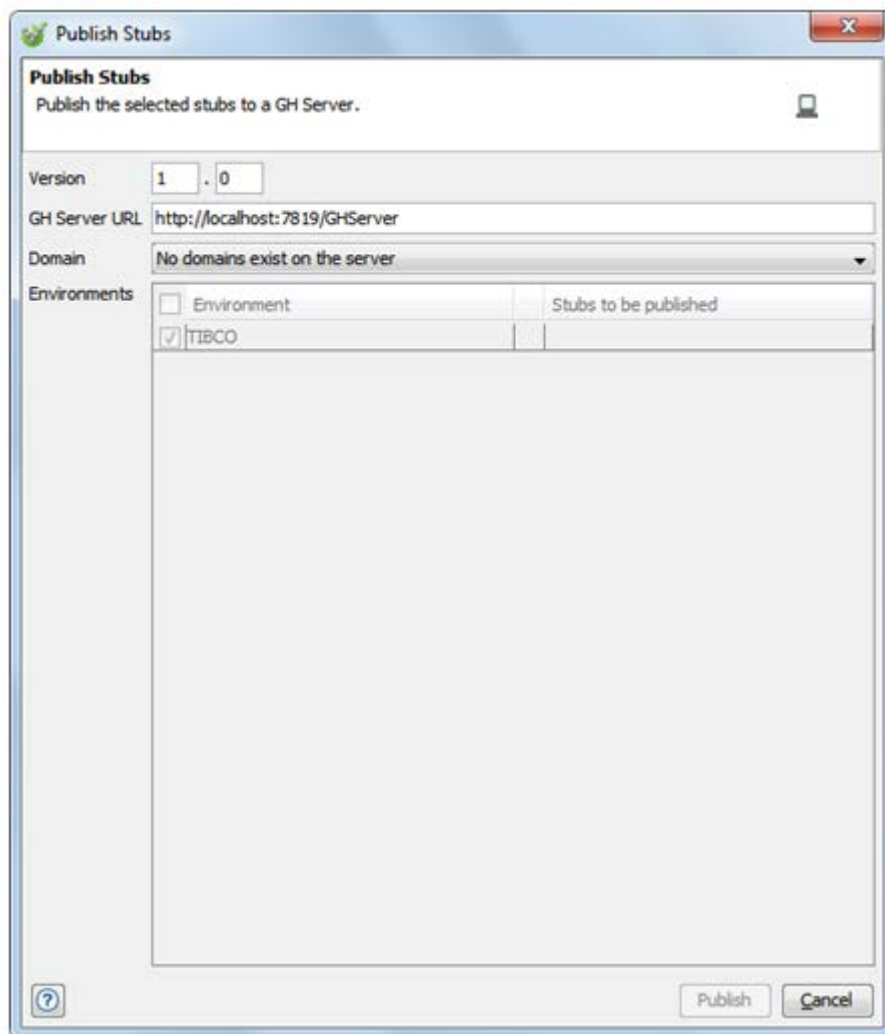
1. In Rational Integration Tester's Test Factory perspective, right-click the **Stubs** folder that contains the stub that you want to publish and click **Publish Stubs...** on the shortcut menu.

Alternatively, right-click the **operation or component** that contains the stub that you want to publish and click **Publish Stubs...** on the shortcut menu.

You cannot publish a single stub by right-clicking it. This is because a stub must be published from a container (that is, an operation, component, or the logical root of the tree) and not from stub itself. All stubs that are within the scope of the selected **Stubs** folder or operation or component or logical root (whichever is applicable) will be published.

**NOTE:** If the operation or component or logical root selected does not contain any stubs, an error message is displayed.

The Publish Stubs dialog box is displayed.



2. In the **Version** fields, enter a version number (an integer) for the stub or set of stubs if more than one was selected for publication.

**NOTE:** A stub's version number assists Rational Test Control Panel users when they are selecting the stubs that they want to run.

3. Edit (if necessary) the **Rational Test Control Panel URL** field to the URL of the Rational Test Control Panel instance to which you want to publish the selected stub(s).

**NOTE:** The default value in the **Rational Test Control Panel URL** field is determined by the URL specified on the **Server Settings** tab on

---

Rational Integration Tester's Project Settings dialog box, which is opened by clicking **Project > Project Settings** on the menu bar.

4. In the **Domain** list, click the domain where you want to publish the selected stub(s).

A domain represents a logical grouping of related systems that are part of a real business project and it is the basic unit of management within Rational Test Virtualization Server.

**NOTE:** The default value in the **Domain** list is determined by the default domain specified in the **Domain** list on the **Server Settings** tab on Rational Integration Tester's Project Settings dialog box.

**NOTE:** The **Domain** list will display error messages if a connection cannot be established to the specified Rational Test Control Panel instance. You will not be able to publish any stubs to the specified Rational Test Control Panel instance unless and until the connection to the instance is re-established.

**Optional:** To create a new domain:

- In the **Domain** list, click **Create new domain**.

The Create a new domain dialog box is displayed.

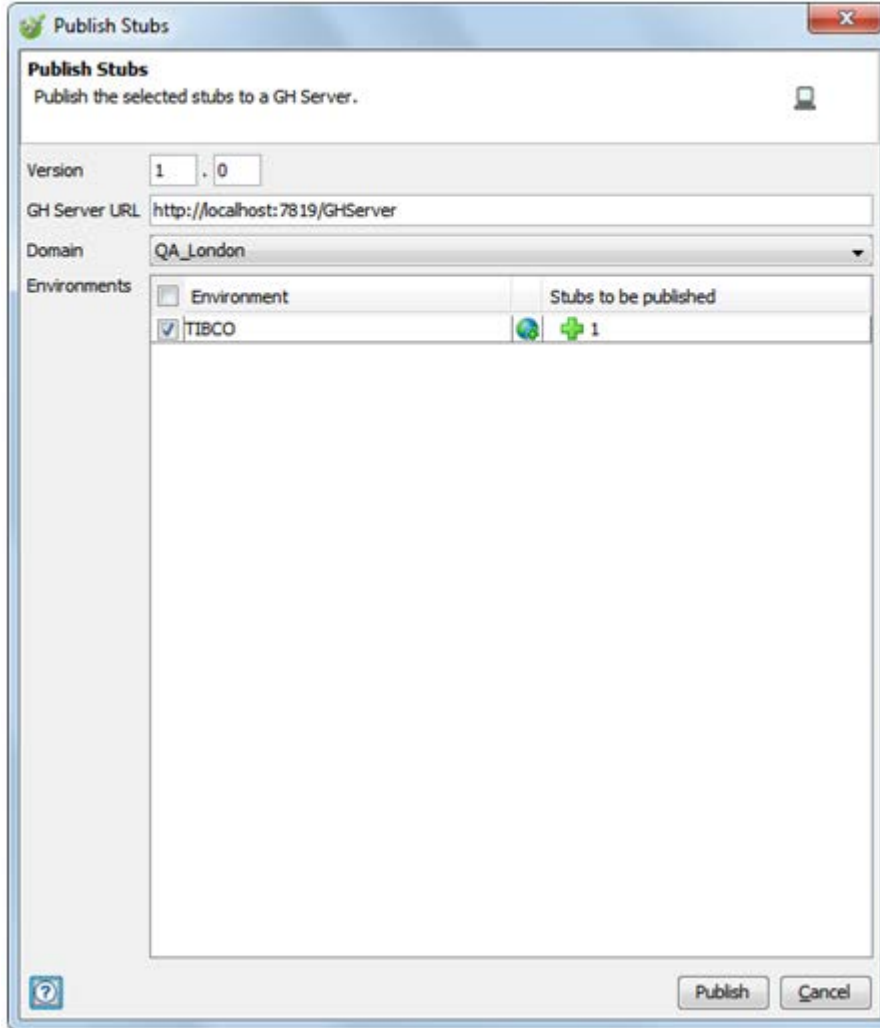


- Click **OK** on the Create a new domain dialog box to close it.

The new domain is displayed in the **Domain** list on the Publish Stubs dialog box.

**NOTE:** If you are a Rational Test Control Panel administrator, you can use Rational Test Control Panel to create and configure new domains. For information about this, refer to *IBM Rational Test Control Panel System Administration Guide*.

5. On the **Environments** window, select the check box(es) of the environment or environments where you want to publish the selected stub(s).








An environment enables Rational Integration Tester and Rational Test Virtualization Server users to define groups of variables or tags that can be used in both tests and transport definitions.

**NOTE:** By default, the check box of the currently selected environment in Rational Integration Tester is selected.

---

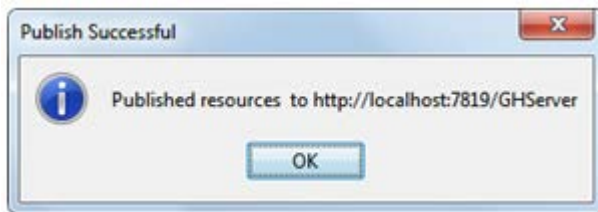
The following table describes the icons that can be displayed on the **Environments** window.

Icon	Description
	The specified environment will be created on the specified Rational Test Control Panel instance.
	The specified environment already exists on the specified Rational Test Control Panel instance.
 "N"	"N" number of new stubs will be published ("created") to the specified Rational Test Control Panel instance.
 "N"	"N" number of existing stubs will be republished ("updated") at the same version level to the specified Rational Test Control Panel instance.
 "N"	"N" number of new versions of existing stubs will be published ("replaced") to the specified Rational Test Control Panel instance.

6. Click **Publish**.

The selected stub(s) is (are) published to the specified Rational Test Control Panel instance.

A status message is displayed to confirm this.



7. Click **OK**.



---

### 5.1.1 Verifying Publication of Stubs

To verify that a stub (or collection of stubs) has (have) been published to a specific Rational Test Control Panel instance:

1. Log into the specified Rational Test Control Panel instance.

**NOTE:** For a brief overview of Rational Test Control Panel, refer to [Using Rational Test Control Panel](#).

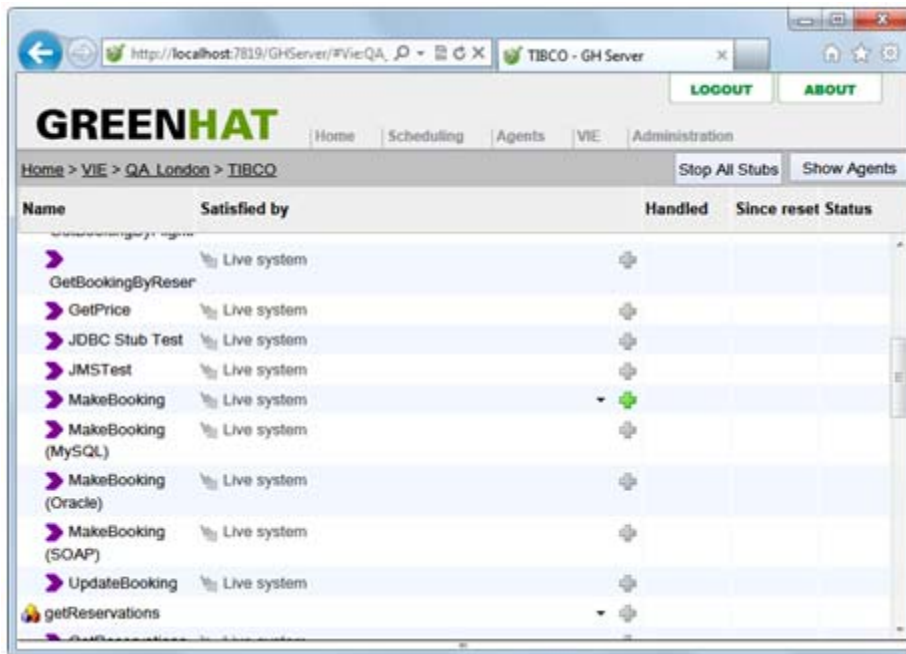
2. Click the **VIE** icon or navigation link.
3. Click the specified domain and environment.



4. Click **View Dashboard**.

---

The selected stub(s) should be displayed on the VIE Dashboard page.



**NOTE:** For information about cancelling the deployment of a stub, refer to [Cancelling Startup](#).

---

## 5.2 Running Stubs

The following sections describe how to start and stop stubs.

### 5.2.1 Starting Stubs

Both Rational Test Control Panel and Rational Integration Tester can be used to start stubs. There are some limitations on stubs if they are started by Rational Integration Tester. The following table describes those limitations.

---

<b>If a stub is published to Rational Test Control Panel and started by Rational Test Control Panel...</b>	<b>If a stub is started by Rational Integration Tester...</b>
There are no running time limits on the stub.	It will run for only <b>five minutes</b> unless the associated test scenario requires it to run for a longer period.
There are no transactions per second (TPS) limits on the stub.	There is a limit of <b>one TPS</b> on the stub.

---

The following sections describe how to use Rational Test Control Panel and Rational Integration Tester to start stubs.

#### 5.2.1.1 Rational Test Control Panel Method

Rational Test Control Panel's VIE Dashboard page enables you to control stubs that have been created and published from Rational Integration Tester, so you do not need to use (or to have access to) Rational Integration Tester to start them.

The VIE Dashboard page also enables you to:

- Override a stub's tag setting.
- Control a stub's behaviour configuration.
- Adjust a stub's response time.

To start a stub:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **VIE** icon or navigation link.

The **VIE** page is displayed.

3. Click the relevant domain and environment.

- 
- Click **View Dashboard**.

The VIE Dashboard page is displayed.

- Under the **Satisfied by** column, click the stub's **plus** button (+).

**NOTE:** If the **plus** button is unavailable, there are no available stubs for the selected operation.

The Start Stub dialog box is displayed on the VIE Dashboard page.

The Start Stub dialog box is displayed. It features a table with columns for Stub, Version, and Documentation. The first row shows 'makeBookingStub' with version '1.0'. The Documentation column contains details about the stub's lifecycle: Published (27-Apr-2012 12:47:23), Updated (26-Apr-2012 18:19:55), and Created (26-Apr-2012 17:08:08). Below the table are expandable sections for Configuration, Agents, and Response time. A Stub label field shows 'makeBookingStub 1.0'. At the bottom are 'Start stub' and 'Cancel' buttons.

Stub	Version	Documentation
makeBookingStub	1.0	<b>Published</b> 27-Apr-2012 12:47:23 By <b>Updated</b> 26-Apr-2012 18:19:55 By <b>Created</b> 26-Apr-2012 17:08:08 By

▸ Configuration

▸ Agents

▸ Response time

Stub label  
makeBookingStub 1.0

Start stub Cancel

---

The following table describes the controls on the Start Stub dialog box.

Control	Description
Stub (page)	<p>This enables you to select the stub, including the stub version, that you want to start.</p> <p>For each stub listed on the page, the <b>Documentation</b> column displays any text entered for that stub on the <b>Documentation</b> tab of the Stub Editor in Rational Integration Tester.</p> <p>After selecting a stub, you can modify its properties by using the other pages on the dialog box.</p> <p><b>NOTE:</b> For database stubs, only one stub for each database can be run at any given time.</p>
Configuration (page)	<p>This enables you to override the default input tag values (for each specified behaviour) that were published with the currently selected stub.</p> <p>Under the <b>Input tags</b> column, each behaviour specified for the stub is listed by name.</p> <p>Clicking a behaviour displays its default input tag values. If you want to override an input tag's default value, click the behaviour and then click the tag and enter a new value under <b>Value</b>. If you want to force the value of the tag to be null instead of being blank, select the <b>Null?</b> check box.</p> <p>Clicking <b>Logging</b> enables you to modify the currently selected stub's log level, which determines how much logging will be written to the project results database. By default, a stub uses the log level that was published with it.</p> <p>Log level options are as follows:</p> <ul style="list-style-type: none"><li>• Default (as defined in the stub)</li><li>• None</li><li>• Normal</li><li>• Debug</li></ul>
Agents (page)	<p>This enables you to select the agents where the currently selected sub will run after it is started.</p> <p>The available agents are displayed, including host name, status, software version number, and system statistics. To select an agent, select its check box.</p> <p><b>NOTE:</b> If the currently selected stub is a database stub, it can run on only one agent at any given time.</p>

---

---

Control	Description
Response time (page)	<p>This enables you to adjust the currently selected stub's response time by entering (in milliseconds) a fixed delay or a delay distribution with minimum and maximum delay times.</p> <p>Delay distributions options are as follows:</p> <ul style="list-style-type: none"><li>• Fixed</li><li>• Uniform</li><li>• Gaussian</li></ul> <p><b>NOTE:</b> After a stub has started, its response time can be modified while it is running (for information about this, refer to <a href="#">Changing Response Times</a>).</p>
Stub label (field)	<p>This is an optional field that enables you to enter a custom label for the currently selected stub so that you distinguish multiple running instances of the stub from each other.</p>

---

- After selecting a stub on the Start Stub dialog and making any desired changes to the stub, clicking **Start Stub** on the dialog box creates a new row under the currently selected operation on the VIE Dashboard page, and `Deploying...` is displayed under the **Status** column.

**NOTE:** For information about viewing stub error messages and stub logs, refer to [Troubleshooting](#).

### Cancelling Startup

If a stub does not deploy reasonably quickly, or if an agent is not operating correctly, you can cancel the deployment of the stub.

To cancel the deployment of a stub:

- Log into Rational Test Control Panel.  
Rational Test Control Panel's application window is displayed.
- Click the **VIE** icon or navigation link.  
The **VIE** page is displayed.
- Click the relevant domain and environment and then click **View Dashboard**.  
The VIE Dashboard page is displayed.
- Under the **Satisfied by** column, click the stub's **arrow** button (↗) and then click **Cancel deployment** on the shortcut menu.



The deployment of the selected stub is cancelled.

### 5.2.1.2 Rational Integration Tester Method

Rational Integration Tester can be used to start stubs but Rational Test Control Panel provides more control over the startup of a stub or a set of stubs.

To start a stub:

1. In Rational Integration Tester's Test Lab perspective, find the stub that you want to start.
2. Right-click the stub and click **Run** on the shortcut menu.

Alternatively, select the stub and click **Run** on the Task Monitor window's toolbar.

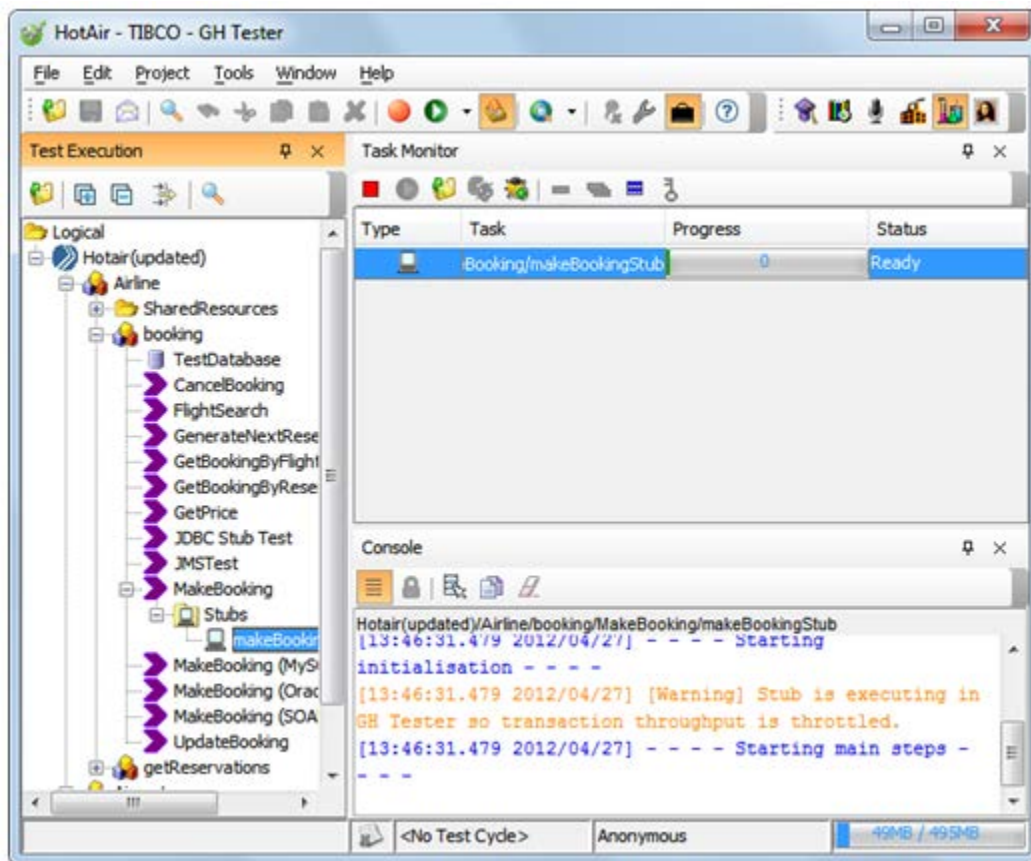
**NOTE:** If you want to run all stubs for a particular operation, right-click the operation's **Stubs** folder and then click **Run All > Stub** on the shortcut menu.

Starting a message-based stub causes the following to happen:

- The Test Lab perspective's Console window displays progress of the stub's startup and messages from the stub as it runs. If the stub's log level has been set to **Debug**, you will be able to view detailed information about what the stub is doing. (For information about setting a stub's log level, refer to [Rational Test](#)

---

Control Panel Method.)



- The stub will start listening on queues/ports.
- If proxies are being used, traffic will be routed to the stub automatically.

**NOTE:** For information about restrictions on stubs started by Rational Integration Tester, refer to [Starting Stubs](#).

Starting a database stub causes the following to happen:

- The data contained in the stub is loaded into the simulation database.
- The Rational Integration Tester JDBC proxy is switched to simulate mode, which affects the system under test but not Rational Integration Tester, and all interactions between the system under test and the live database are directed to the stub. (For information about simulate mode, refer to [Rational Integration Tester JDBC Proxy Modes](#).)



- 
- The Test Lab perspective's Console window displays progress of the stub's startup.

## 5.2.2 Stopping Stubs

The following sections describe how to Use Rational Test Control Panel and Rational Integration Tester stop stubs.

### 5.2.2.1 Rational Test Control Panel Method

Rational Test Control Panel provides several different methods of stopping individual stubs and multiple stubs. The following table describes these methods.

To stop...	Do this...
A stub running on an agent (at agent level)	<p>If using Rational Test Control Panel's VIE Dashboard page:</p> <ol style="list-style-type: none"> <li>1. Click <b>Show Agents</b> (if necessary). A pop-up window is displayed for each agent that is running (in the selected domain/environment, if applicable).</li> <li>2. On the pop-up window of the agent in which you are interested, if the stub that you want to stop is displayed, click the <b>Stop</b> button (⏹) next to the stub's name.</li> </ol> <p>Alternatively, if using Rational Test Control Panel's <b>Agents</b> page:</p> <ol style="list-style-type: none"> <li>1. Click the <b>magnifying glass</b> button (🔍) of an engine. The Agent deployed projects and stubs pop-up window is displayed.</li> <li>2. On the pop-up window of the agent in which you are interested, if the stub that you want to stop is displayed, click the <b>Stop</b> button (⏹) next to the stub's name.</li> </ol>
A stub (on the VIE Dashboard page)	<ol style="list-style-type: none"> <li>1. Under the <b>Satisfied by</b> column, click the stub's <b>arrow</b> button (↕).</li> <li>2. Click <b>Stop stub</b> on the shortcut menu.</li> </ol>
All stubs at operation level (on the VIE Dashboard page)	<ol style="list-style-type: none"> <li>1. Under the <b>Satisfied by</b> column, click the operation's <b>arrow</b> button (↕).</li> <li>2. Click <b>Stop all stubs</b> on the shortcut menu.</li> </ol>
All stubs at component level (on the VIE Dashboard page)	<ol style="list-style-type: none"> <li>1. Under the <b>Satisfied by</b> column, click the component's <b>arrow</b> button (↕).</li> <li>2. Click <b>Stop all stubs</b> on the shortcut menu.</li> </ol>
All stubs for the selected domain/environment (on the VIE Dashboard page)	Click <b>Stop All Stubs</b> (next to <b>Show Agents</b> ).

---

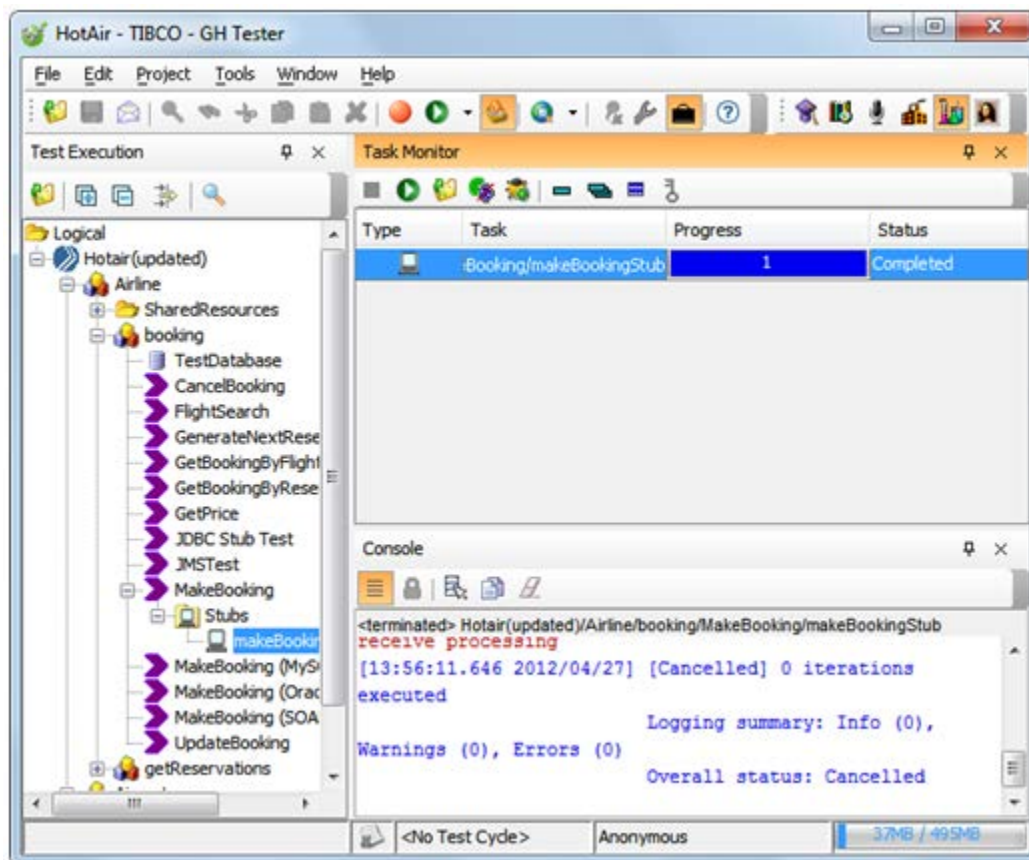
---

### 5.2.2.2 Rational Integration Tester Method

To stop a stub:

1. In Rational Integration Tester's Test Lab perspective, find the stub that you want to stop.
2. Select the stub and click **Stop** on the Task Monitor window's toolbar.

Alternatively, select the stub and click **Stop** on the Task Monitor's toolbar.



Stopping a message-based stub causes the following to happen:

- If traffic was being routed automatically routed to the stub by means of a Rational Integration Tester Proxy, the proxy will be configured to resume sending messages to the live system.

Stopping a database stub causes the following to happen:

- Rational Integration Tester instructs each proxy to return to the state it was in before the stub was started.

- 
- The simulation schema is cleared.

**NOTE:** Stopping a persistent stub updates the stub with a fresh snapshot of data. A non-persistent stub is not updated and, if started again, it will revert to its original state. If there is no connection to the database, exceptions are recorded in a log. In Rational Test Virtualization Server, stubs cannot be persistent.

### 5.2.3 Modifying Running Stubs

If you are using Rational Integration Tester to run a stub, you can modify the stub while it is running.

To modify a running stub:

1. In Rational Integration Tester's Test Factory perspective, double-click the stub to open it in the Stub Editor.
2. Use the Stub Editor to make and save any desired changes to the stub. (For information about using the Stub Editor, refer to [Modifying Message-Based Stubs](#).)
3. Open the Test Lab perspective.

The Task Monitor window shows that the modified stub was stopped and restarted by Rational Integration Tester.

### 5.2.4 Controlling Running Stubs

After a stub is running, the following information is displayed on the VIE Dashboard page:

- **Ready** is displayed under the **Status** column. (If **Error** is displayed, refer to [Stopping Stubs](#).)
- The number of requests that the stub has handled is displayed under the **Handled** column.
- The number of requests that the stub has handled since the last metrics reset is displayed under the **Since reset** column.

Apart from stopping the stub (for information about this, refer to [Stopping Stubs](#)), you can also reset its metrics, change its response times, and cancel its deployment.

The following sections describe these actions.

---

#### 5.2.4.1 Resetting Metrics

To reset the number of requests that a running stub has handled:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **VIE** icon or navigation link.

The **VIE** page is displayed.

3. Click the relevant domain and environment and then click **View Dashboard**.

The VIE Dashboard page is displayed.

4. Under the **Satisfied by** column, click the stub's **arrow** button (↗) and then click **Reset metrics** on the shortcut menu.

The value displayed under **Since reset** column for the selected stub is reset to 0.

#### 5.2.4.2 Changing Response Times

To change the response time of a running stub:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **VIE** icon or navigation link.

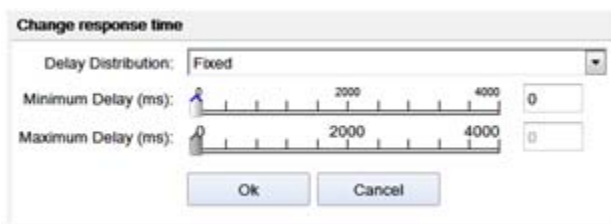
The **VIE** page is displayed.

3. Click the relevant domain and environment and then click **View Dashboard**.

The VIE Dashboard page is displayed.

4. Under the **Satisfied by** column, click the stub's **arrow** button (↗) and then click **Change response time...** on the shortcut menu.

A Change response time dialog box is displayed on the VIE Dashboard page. (For information about the fields on this dialog box, refer to [Starting Stubs](#).)



- 
5. After making any desired changes, click **OK**.

### 5.2.5 Viewing Stub Error Messages

On Rational Test Control Panel's VIE Dashboard page, if **Error** is displayed, resting your mouse pointer over **Error** displays a message about the error.

If an error message includes the “see agent log”, you should refer to the agent's console output and not to any Rational Test Control Panel logs.

### 5.2.6 Viewing Stub Logs

Rational Test Control Panel enables Rational Test Virtualization Server users to view the console output of stubs that have been run in the currently selected domain and environment.

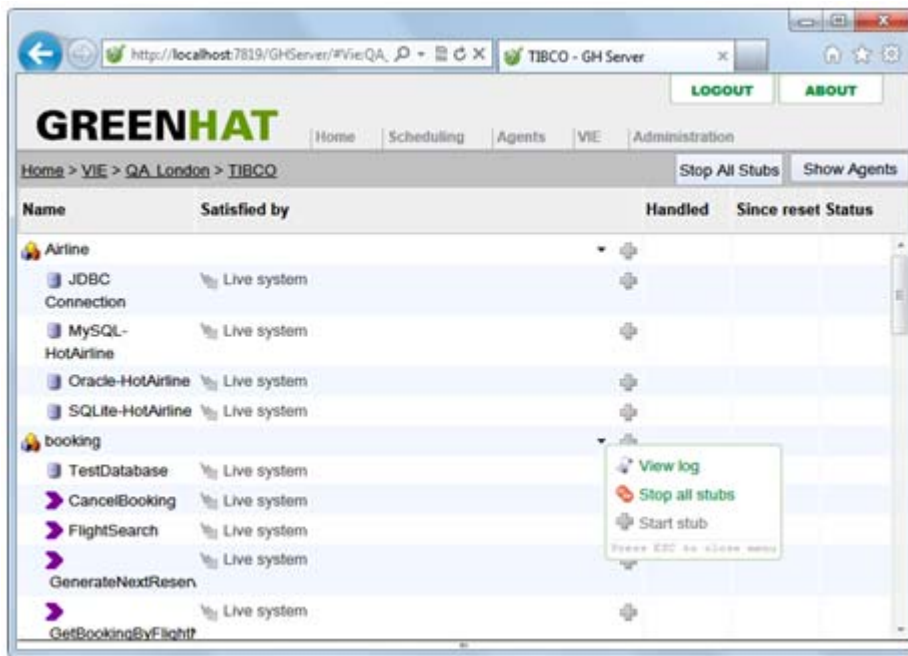
If a domain-level project results database override has been set up for the currently selected domain (for information about this, Rational Test Control Panel administrators should refer to *IBM Rational Test Control Panel System Administration Guide*), only the results database specified in the override is used for displaying the console output of stubs in that domain.

Otherwise, the results databases for all projects published to the currently selected environment that contain the selected stub are used to display the console output of that stub.

When stub logs are viewed, the connection to the specified project results database will be made from the computer running Rational Test Control Panel. When a stub writes its logs, the connection will be made from the computer running the agent upon which the stub is running. (This may not be the computer running Rational Test Control Panel.)

To view a stub's log:

1. On Rational Test Control Panel's VIE Dashboard page, under the **Satisfied by** column, click the **arrow** button ( ) of the **component** (not the operation) that contains the stub.



2. Click **View log** on the shortcut menu.

The operation's stubs log page is displayed.

**NOTE:** If the log page is not displayed or if an empty log page is displayed, refer to [Resolving Stub Log Display Problems](#).

3. **Optional:** Under **Auto scroll**, selecting the **Enable auto scroll on refresh** check box scrolls displays the bottom of the page automatically when new log data arrives.
4. **Optional:** Under **Refresh**, select a refresh interval in the **Interval** list:
  - Clicking **Manual** in the **Interval** list makes the **Refresh** button available, which enables you to refresh the page as often as you wish.
  - Clicking one of the other options in the **Interval** list will refresh the page at set intervals (**10 seconds**, **30 seconds**, **1 minute**, or **5 minutes**).

Under **Warnings**, messages are displayed if results database connection details are incorrect, or if the specified results database is not running, or if its drivers cannot be detected.

Under **Sessions**, any session interactions with the stub are displayed. The same information is also displayed in Rational Integration Tester's Test Lab perspective's Console window.

- 
5. **Optional:** Under **Event types**, selecting and clearing the check boxes of the various event-types under which stub log entries can be classified enables you to filter the log entries displayed on the stubs log page.

# Managing Agents

## Contents

[Viewing All Running Agents  
Registered with Current Rational  
Test Control Panel Instance](#)

[Viewing Agents For a Specific  
Domain/Environment](#)

This chapter describes how to use Rational Test Control Panel to view Rational Integration Tester Agents.

Agents are important components of Rational Test Virtualization Server because they execute stubs that have been published to Rational Test Control Panel.

An agent must be installed on each computer where you want to run stubs and there must be at least one agent in an environment.

Each agent must register with a Rational Test Control Panel instance so that it can accept requests to run stubs.



---

## 6.1 Viewing All Running Agents Registered with Current Rational Test Control Panel Instance

**NOTE:** For background information about agents, refer to *IBM Rational Integration Tester Reference Guide*. For information about deploying agents with Rational Test Control Panel, refer to *IBM Rational Test Control Panel Installation Guide*. For information about installing agents for Rational Test Virtualization Server, refer to *IBM Rational Integration Tester Agent Installation Guide*.

To view **all** agents that are registered with the current Rational Test Control Panel instance and that are currently running:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **Agents** icon or navigation link.

The **Agents** page is displayed.



Any agents registered with the current Rational Test Control Panel instance that are currently running are displayed. Details displayed include IP address, port number,

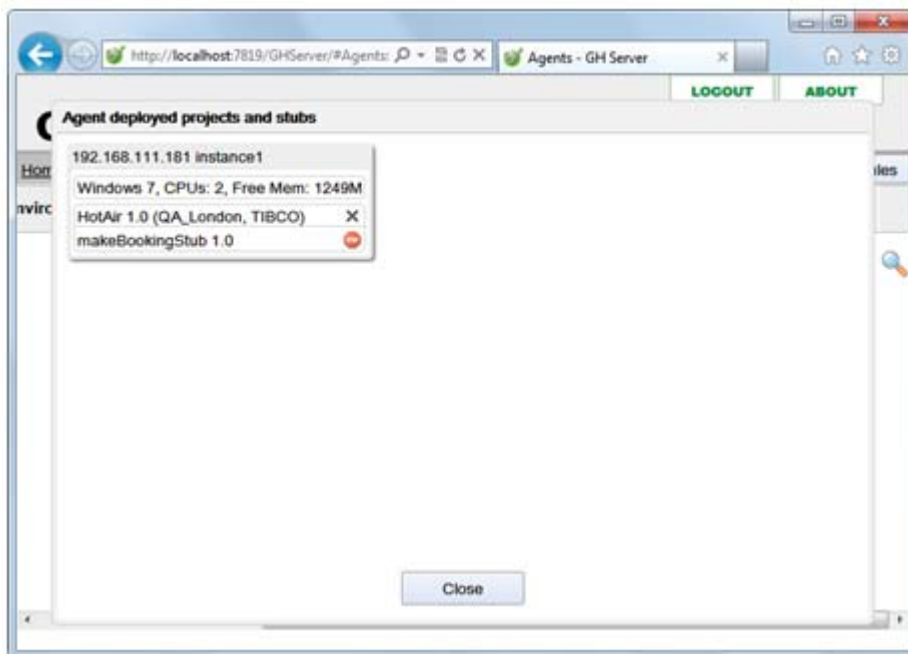
---

domain and environment, software version number, configuration file location, current status, and date and time of last poll.

**NOTE:** A Rational Integration Tester Agent can be registered with only one Rational Test Control Panel instance at any given time.

3. Click the **magnifying glass** button (🔍) of the relevant engine.

The Agent deployed projects and stubs pop-up window is displayed on the **Agents** page.



Details about the selected agent are displayed on the pop-up window, including the IP address of computer where the agent is deployed.

4. Click **Close** to close the Agent deployed projects and stubs pop-up window.

---

## 6.2 Viewing Agents For a Specific Domain/Environment

To view a listing of the agents for a particular domain/environment:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **VIE** icon or navigation link.

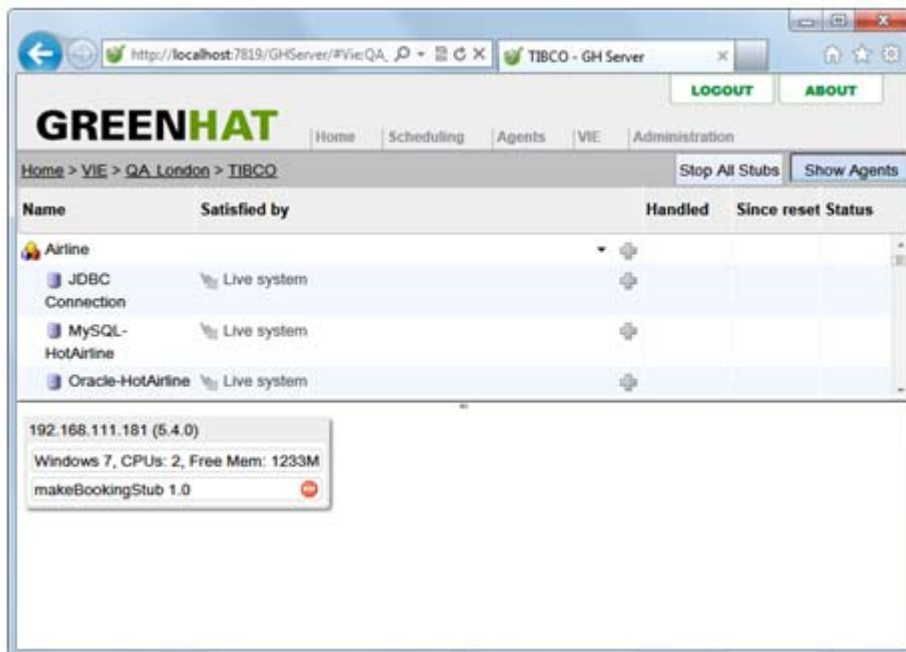
The **VIE** page is displayed.

3. Click the relevant domain and environment and then click **View Dashboard**.

The VIE Dashboard page is displayed.

4. Click **Show Agents**.

The lower half of the VIE Dashboard page displays a pop-up window for each agent for the selected domain/environment that is running.



**NOTE:** Agent console output is not viewable in Rational Test Control Panel, so you must view the console output of an agent on the computer where it is running.

# Troubleshooting

## **Contents**

**Handling Agent Failures**

**Resolving Stub Log Display Problems**

This chapter provides information about how to troubleshoot Rational Test Virtualization Server.

---

## 7.1 Handling Agent Failures

If an agent fails, you can “close” any deployed Rational Test Virtualization Server projects involving that agent.

To close a deployed project:

1. Log into Rational Test Control Panel.

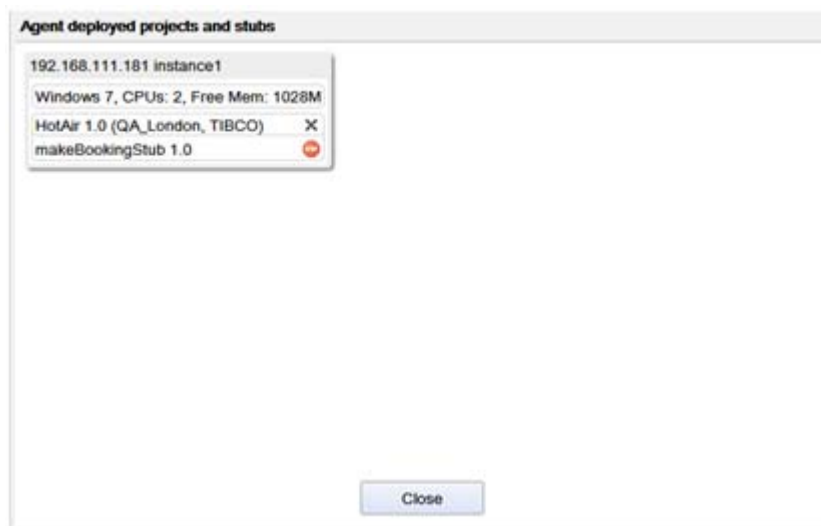
Rational Test Control Panel’s application window is displayed.

2. Click the **Agents** icon or navigation link.

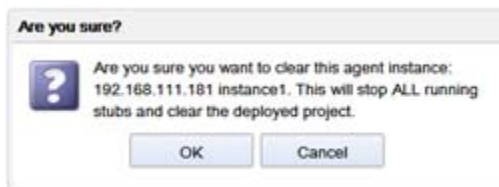
The **Agents** page is displayed.

3. Click the **magnifying glass** button (🔍) of the relevant engine.

The Agent deployed projects and stubs pop-up window is displayed on the **Agents** page.



4. On the pop-up window of the agent that has failed, click the **X** button (✕) next to each project that you wish to close (closing all projects will also stop any stubs that are running for those projects). A confirmation prompt is displayed.



- 
5. Click **OK**.

---

## 7.2 Resolving Stub Log Display Problems

The following table outlines reasons why stub logs might not be displayed and describes how to resolve any problems.

Possible Reason	Resolution
The stub is not logging events.	<p>In Rational Test Control Panel's VIE dashboard (or in Rational Integration Tester), verify that the stub has been configured to <code>Log</code>.</p> <p>Alternatively, if <code>localhost</code> has been configured as the logging URL, logging will not work if the stub is run on a different agent.</p>
The results database for the stub (or to be more precise, the project from which the stub was published) is not reachable by the stub running on the agent.	<p>Ensure that the computer running the stub and the computer running Rational Test Control Panel have the required network connectivity to access the project results database.</p>
The required database driver is missing.	<p>If you are using a MySQL driver, ensure that it has been installed correctly (for information about this, refer to <i>IBM Rational Test Control Panel Installation Guide</i>).</p> <p>If you are using a non-standard JDBC driver, ensure that it is included in the <code>CLASSPATH</code> of the computer where Rational Test Control Panel is installed.</p>

---

# Appendix A: Using the Data Model Editor

## Contents

[Creating Data Models](#)

[Editing Data Models](#)

[Deleting Data Models](#)

In Rational Integration Tester, you can create a data model while creating a data model-driven stub from recorded events (for information about this, refer to [Creating Data Model-Driven Stubs](#)).

Alternatively, you can use the Data Model Editor.

This appendix describes how to use the Data Model Editor to create, modify, and delete data models.



---

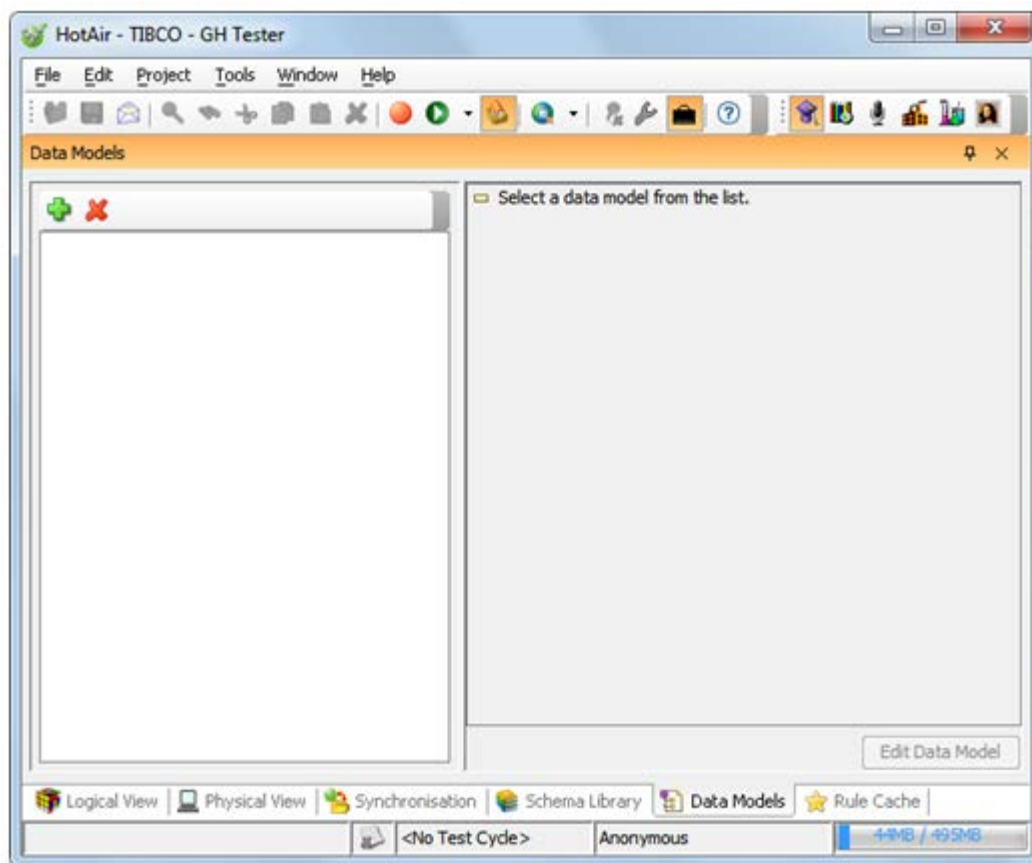
## 8.1 Creating Data Models

A data model enables you to understand the data held by the system that you want to simulate, and it can provide persistent storage of that data.

To create a new data model:

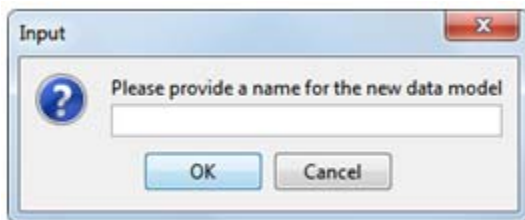
1. Open Rational Integration Tester's Architecture School perspective.
2. Click the **Data Models** tab.

The Data Models window is displayed.



3. Click the **plus** button (+).

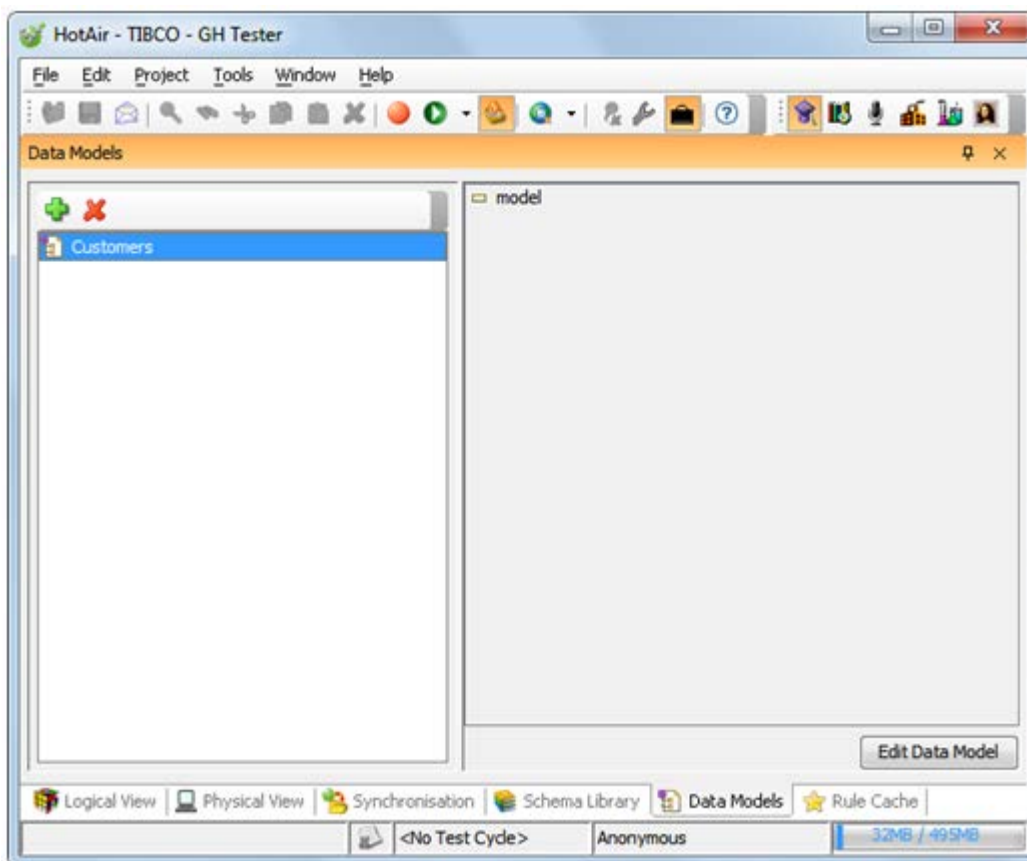
The Input dialog box is displayed.



4. Enter a name for the new data model.
5. Click **OK**.

The name of the new data model is displayed on the upper left of the Data Models window.

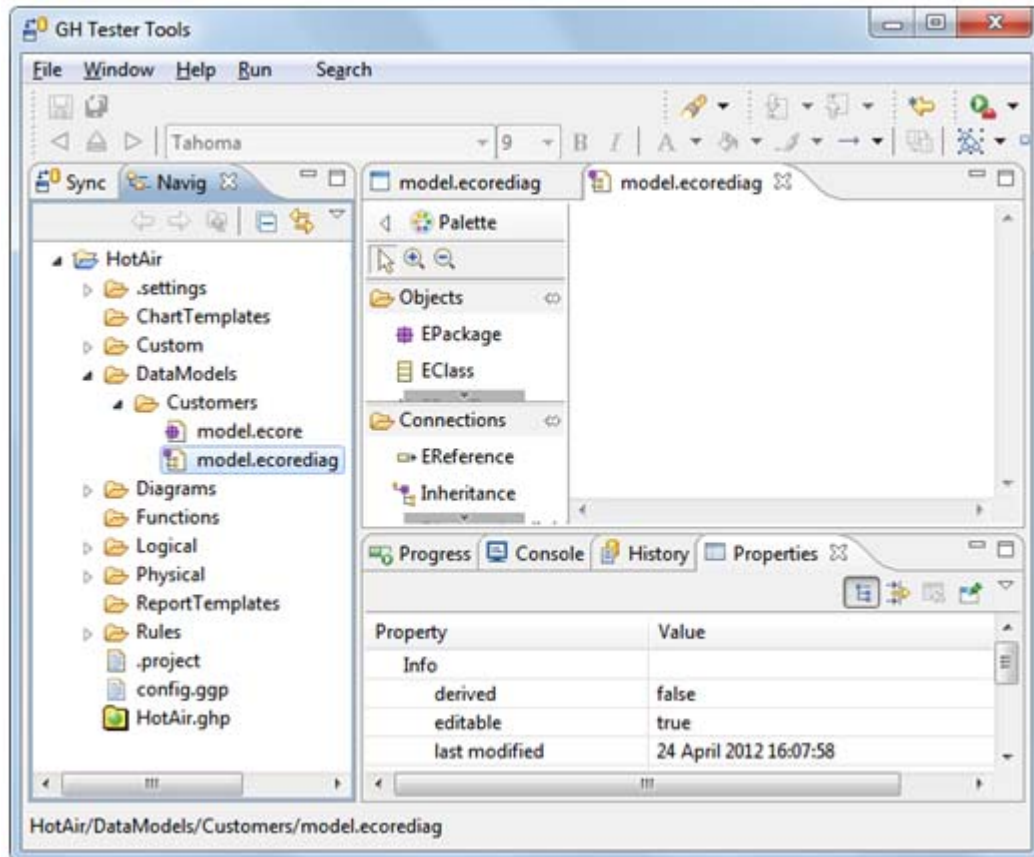
6. Click the new data model.



7. Click **Edit Data Model**.

---

The Rational Integration Tester Tools application is opened and the Data Model Editor is displayed.



The instance data for a data model is maintained within a set of comma-separated value (CSV) files. Whenever the data model editor is used to make changes to a data model, appropriate changes are made to the instance data store. The changes are applied as a result of saving the updated data model diagram. If any additional information is needed, it is requested at this point.

The **Navigator** tab on the left of the Rational Integration Tester Tools application window displays a collection of folders, including a **DataModels** folder, for the currently selected Rational Integration Tester project.

---

The following table describes how to use the palette (on the **model.ecorediag** tab to the left of the blank canvas) to add entities, attributes, and references.

---

To add...	Do this...
<p>An entity.</p> <p>For example, <code>Customer</code>.</p>	<ol style="list-style-type: none"><li>1. Click <b>EClass</b>.</li><li>2. Click anywhere on the blank canvas to insert a new entity.</li><li>3. Right-click the new entity.</li><li>4. Click <b>Show Properties View</b> on the shortcut menu.</li><li>5. In the <b>Name</b> field (on the <b>Properties</b> tab under the canvas), enter a name for the new entity.</li><li>6. Press ENTER (or move to another field on the <b>Properties</b> tab) to apply the new name.</li></ol>
<p>An attribute to an entity.</p> <p>An attribute is a named scalar property of an instance of an entity.</p> <p>For example, an entity <code>Customer</code> might have an attribute <code>emailAddress</code> of a Java <code>java.lang.String</code> type.</p> <p>In general, the type (<b>EType</b>) that you will select in the Data Model Editor will be one of the following:</p> <ul style="list-style-type: none"><li>• <b>EString</b> (equivalent to a Java <code>java.lang.String</code> type)</li><li>• <b>EBoolean</b> (equivalent to a Java primitive <code>boolean</code>)</li><li>• <b>EDate</b> (equivalent to a Java <code>java.util.Date</code> type)</li><li>• <b>EDouble</b> (equivalent to a Java primitive <code>double</code> type)</li><li>• <b>EInt</b> (equivalent to a Java primitive <code>int</code> type)</li><li>• <b>ELong</b> (equivalent to a Java primitive <code>long</code> type)</li></ul>	<ol style="list-style-type: none"><li>1. Click <b>EAttribute</b>.</li><li>2. Click the relevant entity on the canvas.</li><li>3. In the second row of the entity object on the canvas, enter the name of the attribute.</li><li>4. On the <b>Properties</b> tab under the canvas, click the <b>Browse</b> button next to the <b>EType</b> field.</li><li>5. On the Object selection dialog box, select an appropriate type.</li><li>6. Click <b>OK</b>.</li></ol>

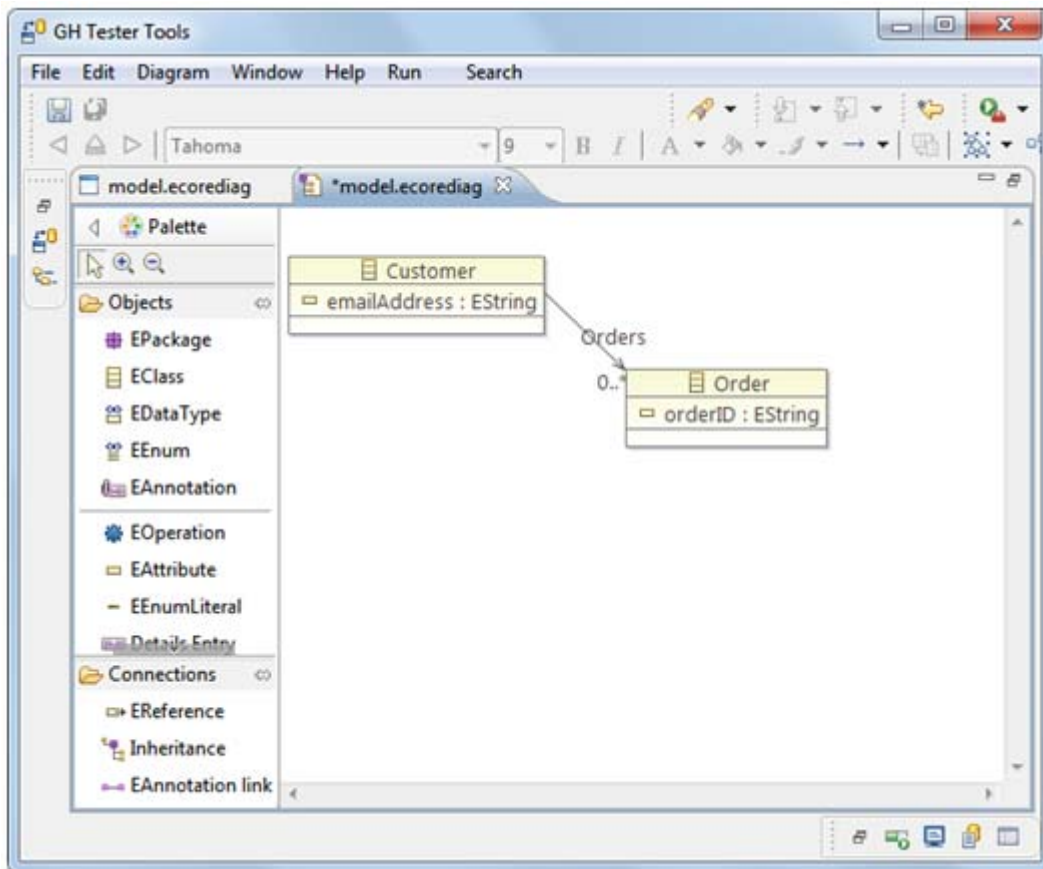
---

---

To add...	Do this...
A reference between entities. A reference indicates a relationship between two entities. For example, there might be a relationship between a customer and the number of orders that that customer has placed	<ol style="list-style-type: none"><li>1. Create the entities on the canvas.</li><li>2. Click <b>EReference</b>.</li><li>3. On the canvas, draw a line from the entity that holds the reference to the entity that is the type of the reference.</li><li>4. Right-click the line.</li><li>5. Click <b>Show Properties View</b> on the shortcut menu.</li><li>6. In the <b>Name</b> field (on the <b>Properties</b> tab under the canvas), enter a name for the reference.</li><li>7. In the <b>Upper Bound</b> field, enter 1 (for a 0..1 reference) or -1 (for a 0..* reference).</li><li>8. Press ENTER (or move to another field on the <b>Properties</b> tab) to apply the new name.</li></ol>

---

After you create a data model, the data is held inside a set of entities, and each entity has a set of attributes.



---

## 8.2 Editing Data Models

Editing a data model involves modifying or deleting components of the model.

**NOTE:** When editing data models, ensure that you click the correct **DataModels** folder on the **Navigator** tab on the Rational Integration Tester Tools application window before making any changes.

The following table describes how to use the Data Model Editor in the Rational Integration Tester Tools application to edit a data model.

To...	Do this...
Rename an entity, attribute, or reference.	<ol style="list-style-type: none"><li>1. On the canvas, right-click the entity, attribute, or reference (as applicable).</li><li>2. Click <b>Show Properties View</b> on the shortcut menu.</li><li>3. In the <b>Name</b> field, modify the name of the entity, attribute, or reference (as applicable).</li><li>4. Press ENTER.</li></ol>
Move an attribute between entities. <b>NOTE:</b> An attribute can be moved from one entity to another. If the model has instance data, the data can be mapped between instances of the affected entities by providing a foreign key mapping from the original entity to the new entity when the Data Model Editor is saved.	<ol style="list-style-type: none"><li>1. On the canvas, drag an attribute from the original entity object and drop it on the second row of new entity object.</li><li>2. Click <b>Save</b> to open the Mapping dialog box, which enables you to specify how instances of one entity can be mapped to another entity.</li><li>3. On the Mapping dialog box, select the <b>Use these mappings for all feature moves from &lt;Source Entity&gt; to &lt;Target Entity&gt;</b> check box if you want to reuse the current mapping for any future attribute moves between the currently selected entities. Alternatively, click <b>No Mapping</b> if you do not want to create a mapping between the currently selected entities.</li></ol>

---

To...	Do this...
Move a reference between entities.	<ol style="list-style-type: none"> <li>1. On the canvas, move the source (non-arrow) end of the reference line to the target entity.</li> <li>2. Click <b>Save</b> to open the Mapping dialog box, which enables you to specify how instances of one entity can be mapped to another entity.</li> <li>3. On the Mapping dialog box, select the <b>Use these mappings for all feature moves from &lt;Source Entity&gt; to &lt;Target Entity&gt;</b> check box if you want to reuse the current mapping for any future attribute moves between the currently selected entities. Alternatively, click <b>No Mapping</b> if you do not want to create a mapping between the currently selected entities.</li> </ol>
Modify the type of a reference.	<ol style="list-style-type: none"> <li>1. On the canvas, move the destination (arrow) end of the reference line to the target entity.</li> <li>2. Click <b>Save</b> to open the Mapping dialog box, which enables you to specify how instances of one entity can be mapped to another entity.</li> <li>3. On the Mapping dialog box, select the <b>Use these mappings for all feature moves from &lt;Source Entity&gt; to &lt;Target Entity&gt;</b> check box if you want to reuse the current mapping for any future attribute moves between the currently selected entities. Alternatively, click <b>No Mapping</b> if you do not want to create a mapping between the currently selected entities.</li> </ol>
Modify the cardinality of a reference.	<ol style="list-style-type: none"> <li>1. On the canvas, right-click the reference.</li> <li>2. Click <b>Show Properties View</b> on the shortcut menu.</li> <li>3. In the <b>Upper Bound</b> field, enter 1 (for a 0..1 reference) or -1 (for a 0..* reference).</li> <li>4. Press ENTER.</li> </ol>



---

---

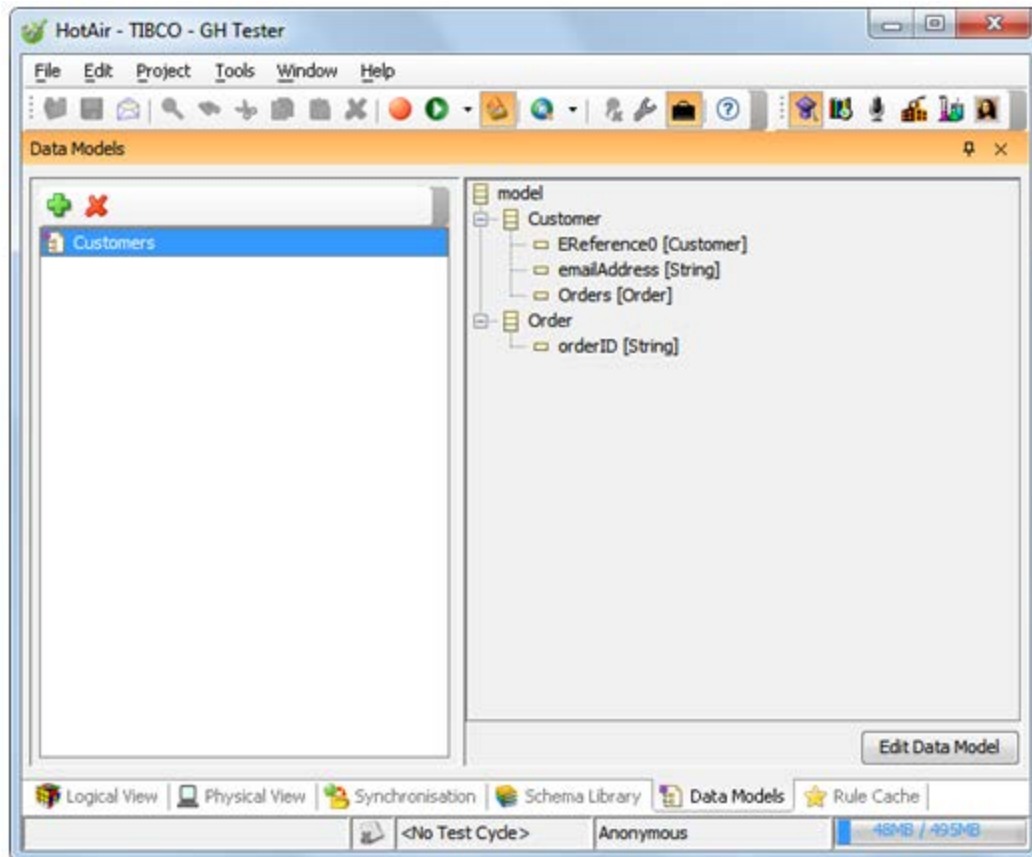
To...	Do this...
Merge two attributes of an entity.	<ol style="list-style-type: none"><li>1. On the canvas, hold CTRL and click the two attributes of an entity that you want to merge.</li><li>2. Click <b>Merge Attributes</b> on the shortcut menu.</li><li>3. On the Merge Attributes dialog box:<ul style="list-style-type: none"><li>• In the <b>Attribute to retain</b> list, click the attribute that you want to retain as the resulting single attribute.</li><li>• In the <b>Precedent attribute</b> list, click the attribute that has the value that will be used in situations where an instance of that type has data for each attribute.</li></ul></li></ol>
Delete an entity, attribute, or reference. <b>NOTE:</b> If you delete an entity, any stored instance data for that entity is also deleted. If you delete an attribute or a reference, all data for that attribute/reference is deleted from instances of the entity that contains that attribute/reference.	<ol style="list-style-type: none"><li>1. On the canvas, right-click the entity, attribute, or reference (as applicable).</li><li>2. Click <b>Delete from Model</b> on the shortcut menu.</li></ol>

---

---

To verify any changes that you have made to a data model:

1. After clicking **Save** in the Data Model Editor, view the Data Models window on Rational Integration Tester's Architecture School perspective.



2. If there is more than one data model listed on the upper left of the Data Models window, click the data model that you have modified in the Data Model Editor to refresh its details on the Data Models window. (You might have to reselect the data model to refresh the window.)

---

## 8.3 Deleting Data Models

If you delete a data model, any resources in Rational Integration Tester that reference that data model will no longer be valid.

To delete a data model:

1. Open Rational Integration Tester's Architecture School perspective.
2. Click the **Data Models** tab.
3. On the upper left of the Data Models window, click the data model that you want to delete.
4. Click the red **X** button (✖).

A confirmation prompt is displayed.

5. Click **Yes**.

The selected data model is deleted.

# Appendix B: Creating Behaviours

## **Contents**

### **[Introduction](#)**

### **[Creating a Behaviour](#)**

### **[Defining Interfaces](#)**

This appendix describes how to create a simple behaviour called “echo” that will expose one function and one event for use by a Rational Test Virtualization Server stub.

For information about how behaviours are used by Rational Test Virtualization Server, refer to [Creating & Modifying Message-Based Stubs](#).

---

## 9.1 Introduction

Behaviours are reusable behavioural units that can expose utility or business logic for use within Rational Test Virtualization Server. They can be combined to build rich virtualized applications with minimal development effort.

A behaviour interacts with a stub in two ways:

- It provides methods that can be used within script actions in the business logic of a stub. This enables the stub to control the behaviour at run time.
- It can raise events to the stub, passing information that tells the stub that the behaviour needs to interact in some way, such as sending a message.

Behaviours are discovered by Rational Test Virtualization Server at run time for use in stubs by means of an Eclipse extension point:

`com.greenhat.tester.api.behaviour` and they have the following four primary elements:

1. The **factoryClass** is a Java class whose package is exported by the behaviour-contributing plugin that implements the `com.greenhat.tester.api.behaviour.BehaviourFactory` interface. The implementation of the class creates and returns an implementation of the behaviour and typically will configure and add listeners to it.
2. The **behaviourInterface** is a Java interface that is an extension of the `com.greenhat.tester.api.behaviour.LifecycleAwareBehaviour` interface whose package is exported by the behaviour-contributing plug-in, which exposes the functions of the behaviour.
3. The **callbackInterface** is a Java interface whose package is exported by the behaviour--contributing plugin and exposes the events exposed as call-backs by the behaviour.
4. The **implementation** of the behaviour is custom code that is typically instantiated and configured by the **factoryClass**, which implements the **behaviourInterface** and makes call-backs on the **callbackInterface**.

---

## 9.2 Creating a Behaviour

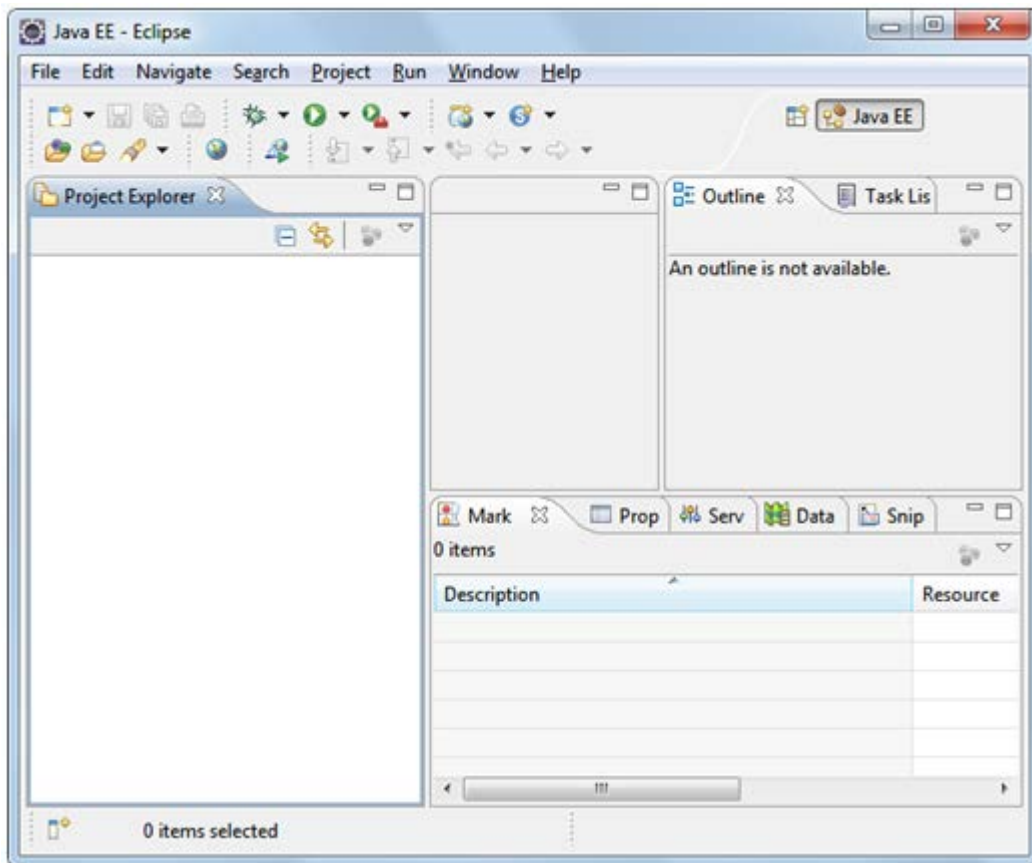
Before creating a behaviour, you will need to:

- Install an Eclipse IDE with the Plug-in Development Environment (PDE). For example, Eclipse for RCP and RAP Developers, which can be downloaded from [www.eclipse.org](http://www.eclipse.org)
- Copy the `com.greenhat.testers.api_<Version Number>.jar` file from `<Rational Integration Tester Installation Directory>\plugins` to `<Eclipse IDE Installation Directory>\plugins` on the target computer.

To create a new behaviour for Rational Test Virtualization Server:

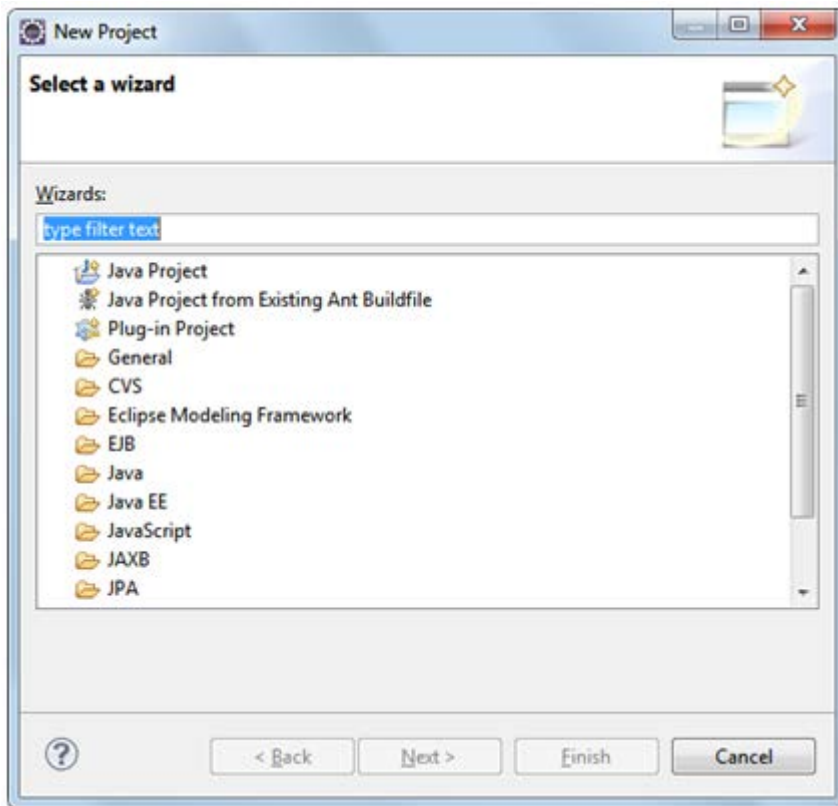
1. Open Eclipse.

The Eclipse application window is displayed.



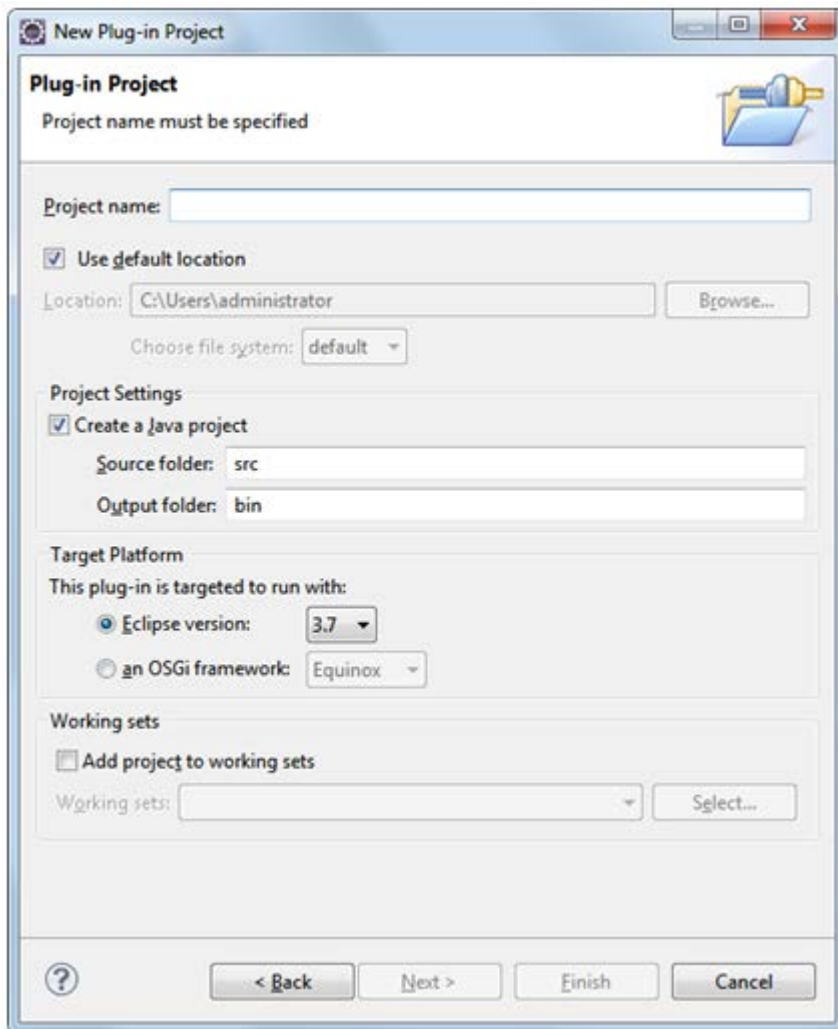
2. Click **File > New> Project**.

The New Project wizard is displayed.



3. Click **Plug-in Project**.
4. Click **Next**.

The first screen of the New Plug-in Project wizard is displayed.



5. In the **Project name** field, enter `com.example.behaviour.echo`.
6. Select the **Use default location** check box.
7. Select the **Create a Java project** check box.
8. In the **Source folder** field, enter `src`.
9. In the **Output folder** field, enter `bin`.
10. Click the **Eclipse version** option button.
11. In the **Eclipse version** list, select the appropriate platform version.
12. Clear the **Add project to working sets** check box.



**New Plug-in Project**

**Plug-in Project**  
Create a new plug-in project

Project name:

☒ Use default location

Location:

Choose file system:

**Project Settings**

☒ Create a Java project

Source folder:

Output folder:

**Target Platform**

This plug-in is targeted to run with:

☒ Eclipse version:

☐ an OSGi framework:

**Working sets**

☐ Add project to working sets

Working sets:

- 
13. Click **Next**.

The second screen of the New Plug-in Project wizard is displayed.

**New Plug-in Project**

**Content**  
Enter the data required to generate the plug-in.

**Properties**

ID:

Version:

Name:

Provider:

Execution Environment:

**Options**

☒ Generate an activator, a Java class that controls the plug-in's life cycle  
Activator:

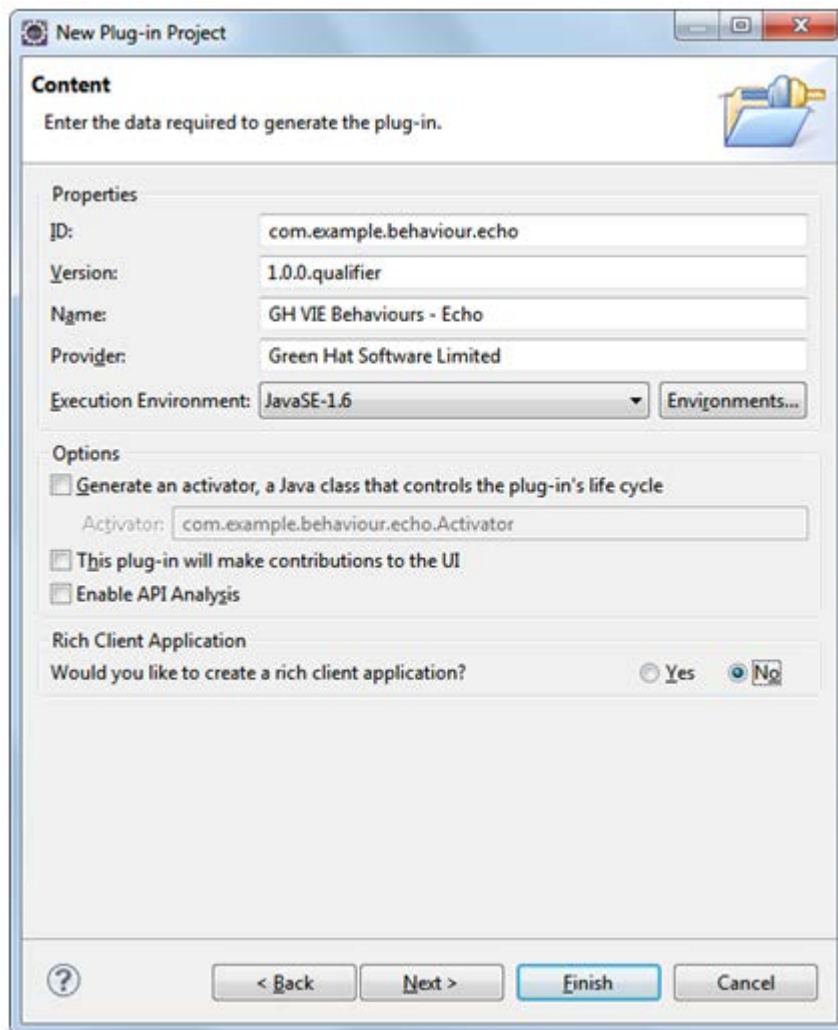
☐ This plug-in will make contributions to the UI

☐ Enable API Analysis

**Rich Client Application**  
Would you like to create a rich client application? ☐ Yes ☒ No

14. In the **ID** field, enter `com.example.behaviour.echo` (if necessary).
15. In the **Version** field, enter `1.0.0 qualifier` (if necessary).
16. In the **Name** field, enter Rational Test Virtualization Server Behaviours - Echo (or an equivalent name).
17. In the **Provider** field, enter IBM Corporation.
18. In the **Execution Environment** field, click **JavaSE-1.6**.
19. Under **Options**, clear all the check boxes.

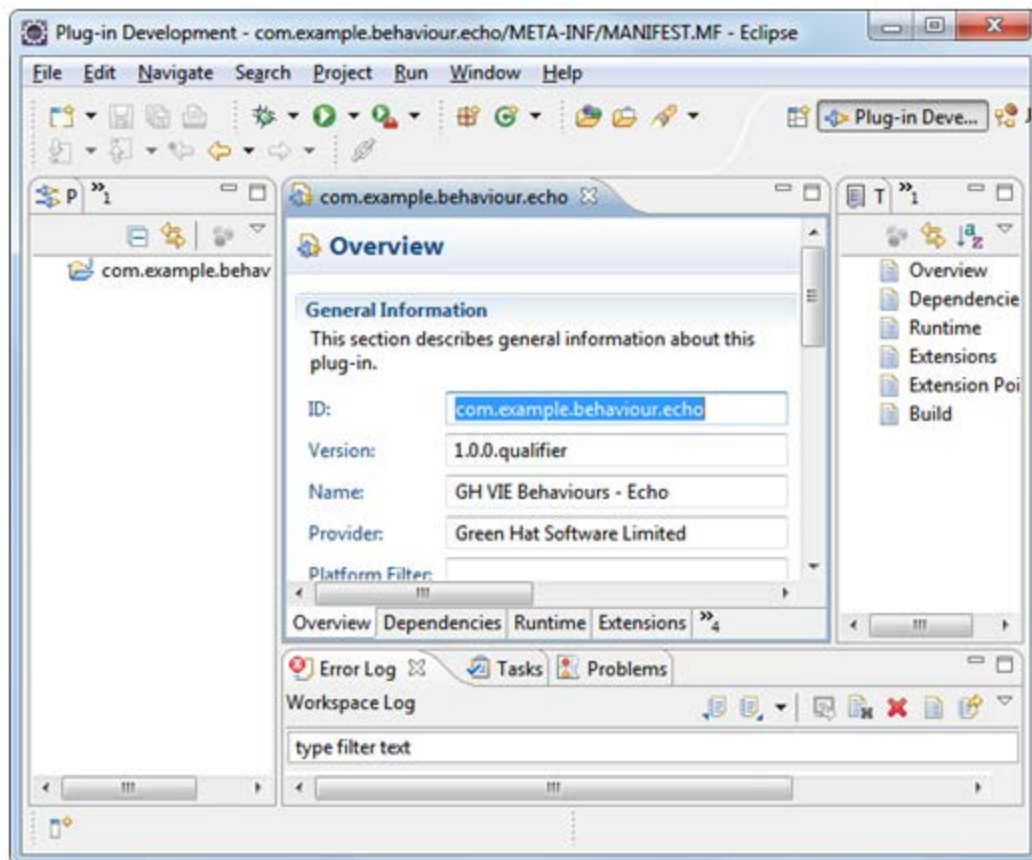
- 
20. Under **Rich Client Application**, click the **No** option button.



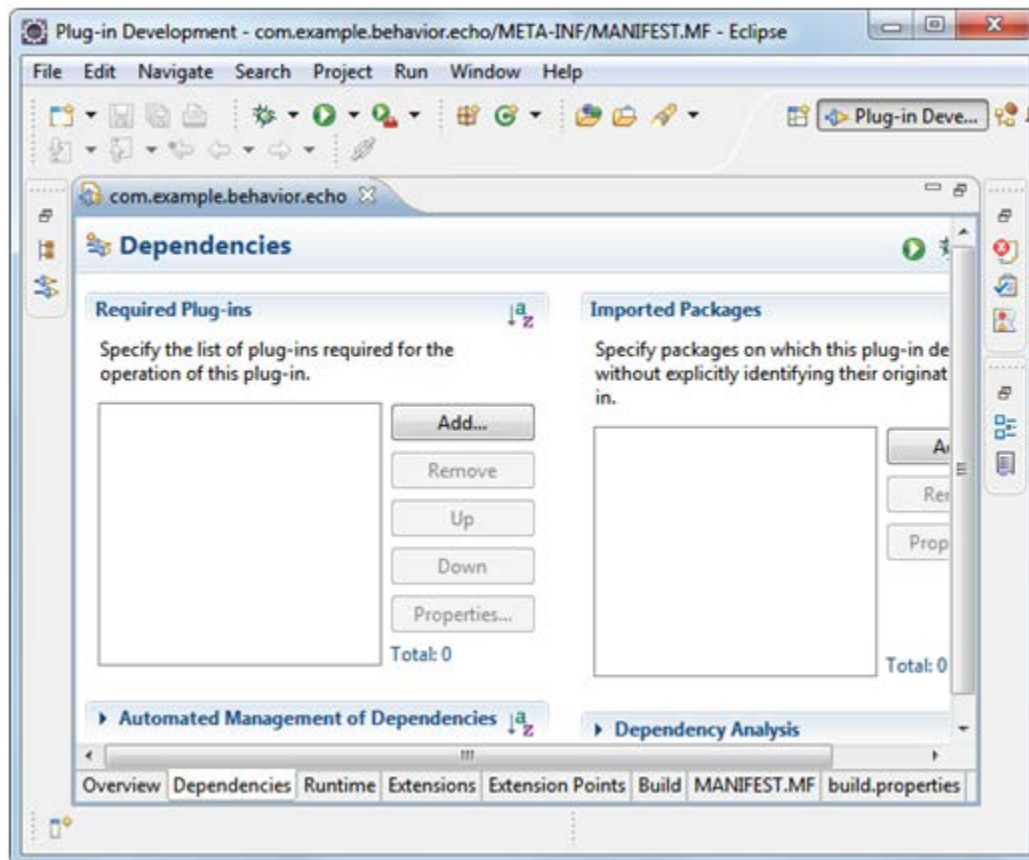
21. Click **Finish**.

**NOTE:** If a message about switching to Eclipse's Plug-in Development perspective is displayed, click **Yes**.

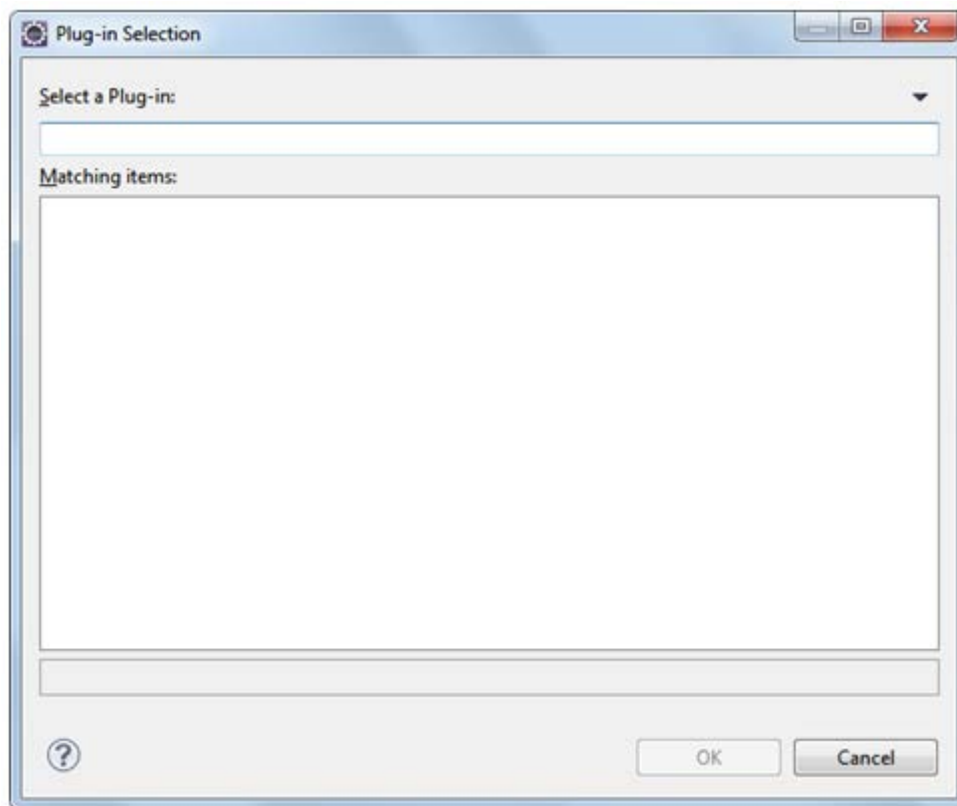
The Manifest Editor for the plug-in is displayed.



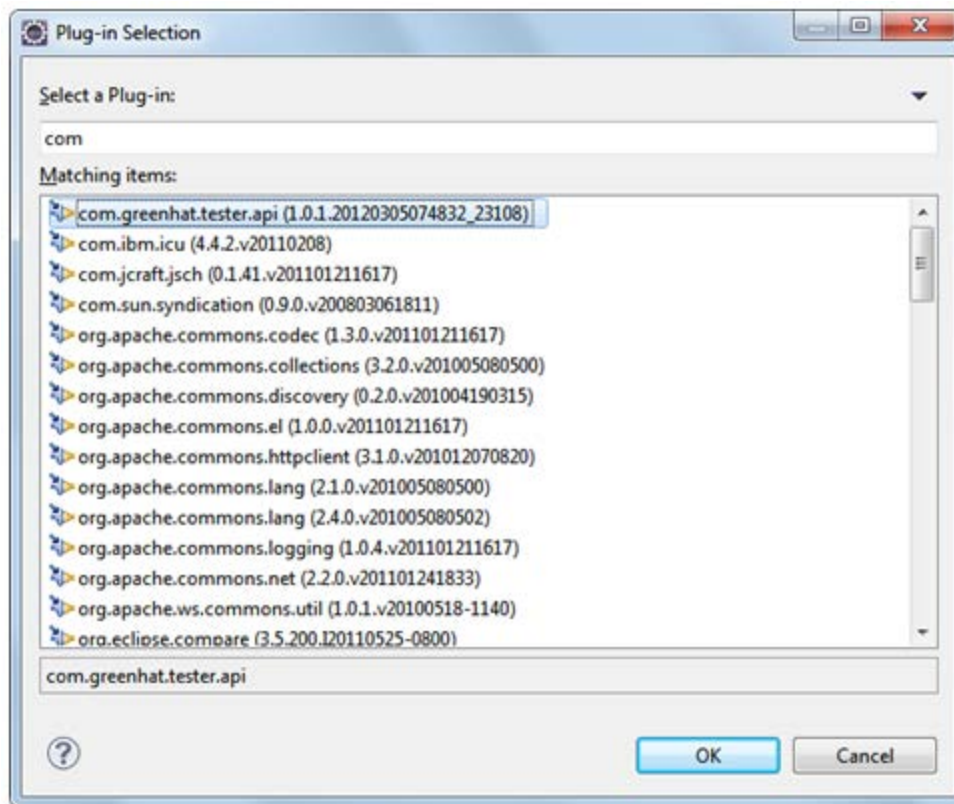
22. On the Manifest Editor window, click the **Dependencies** tab.  
The **Dependencies** tab is displayed.
23. **Optional:** Maximize the Manifest Editor window.



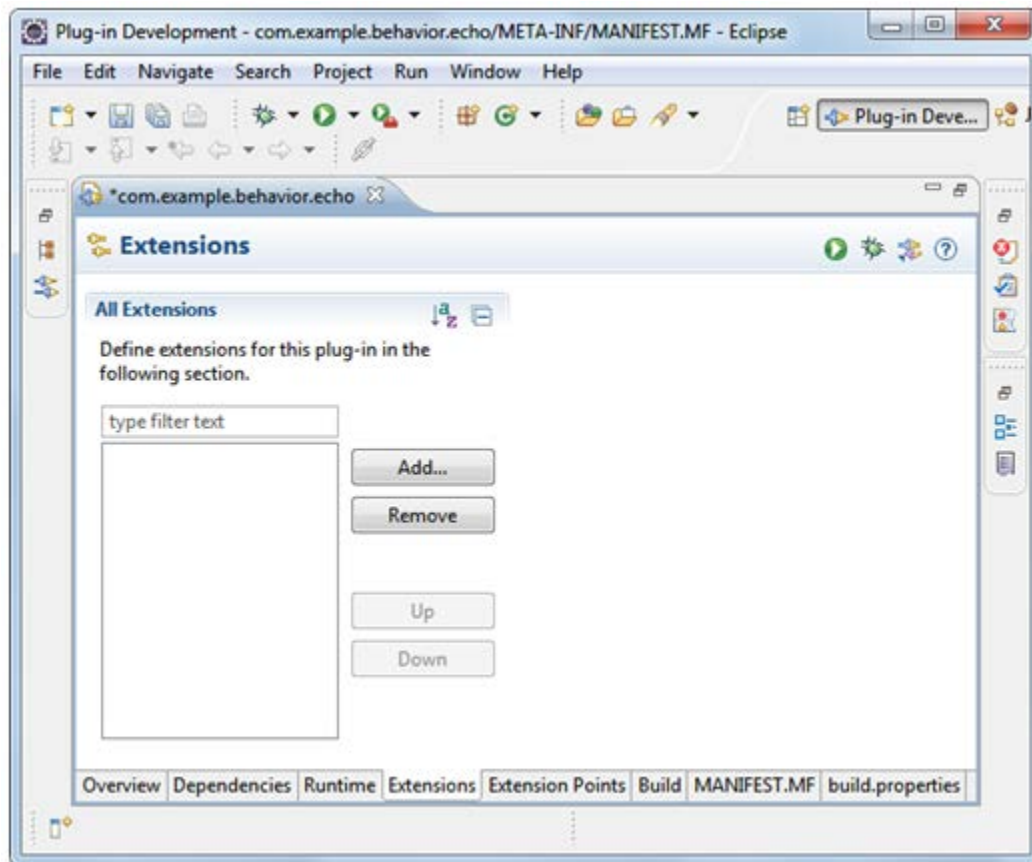
24. Under **Required Plug-ins**, click **Add**.  
The Plug-in Selection dialog box is displayed.



25. In the **Select a Plug-in** field, enter at least one character of the JAR file's name to prompt the dialog box to filter the list of plug-ins under **Matching Items**.
26. Click **com.greenhat.testers.api** (*Version Number*).



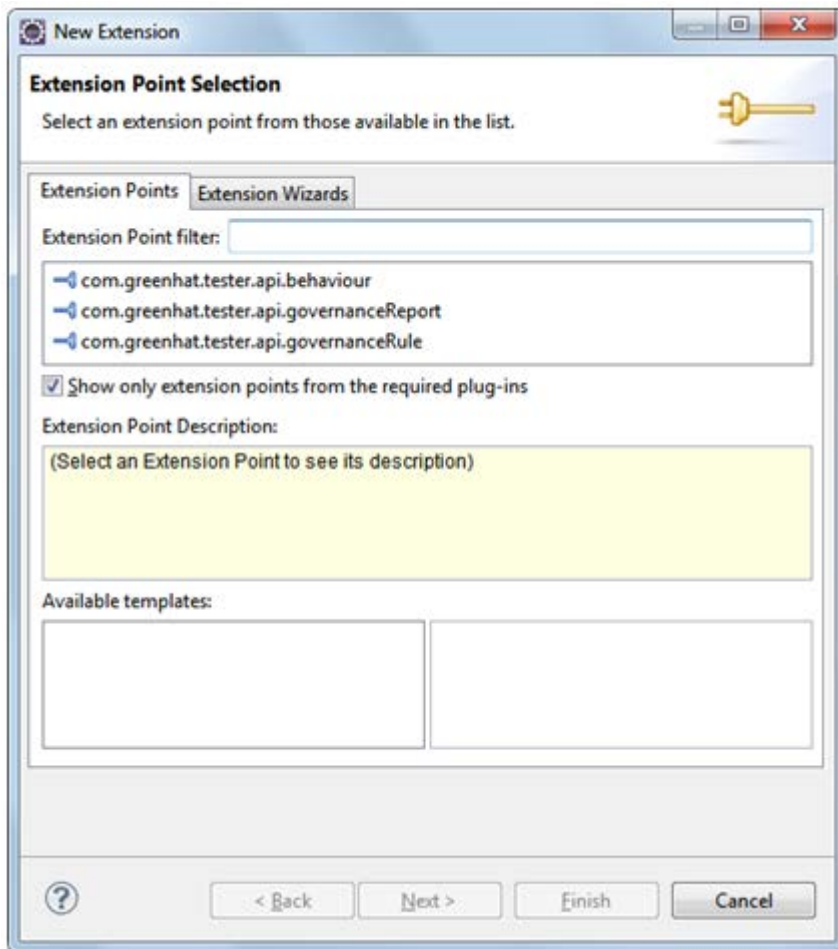
27. Click **OK**.
28. On the Manifest Editor window, click the **Extensions** tab.  
The **Extensions** tab is displayed.



29. Under **All Extensions**, click **Add**.

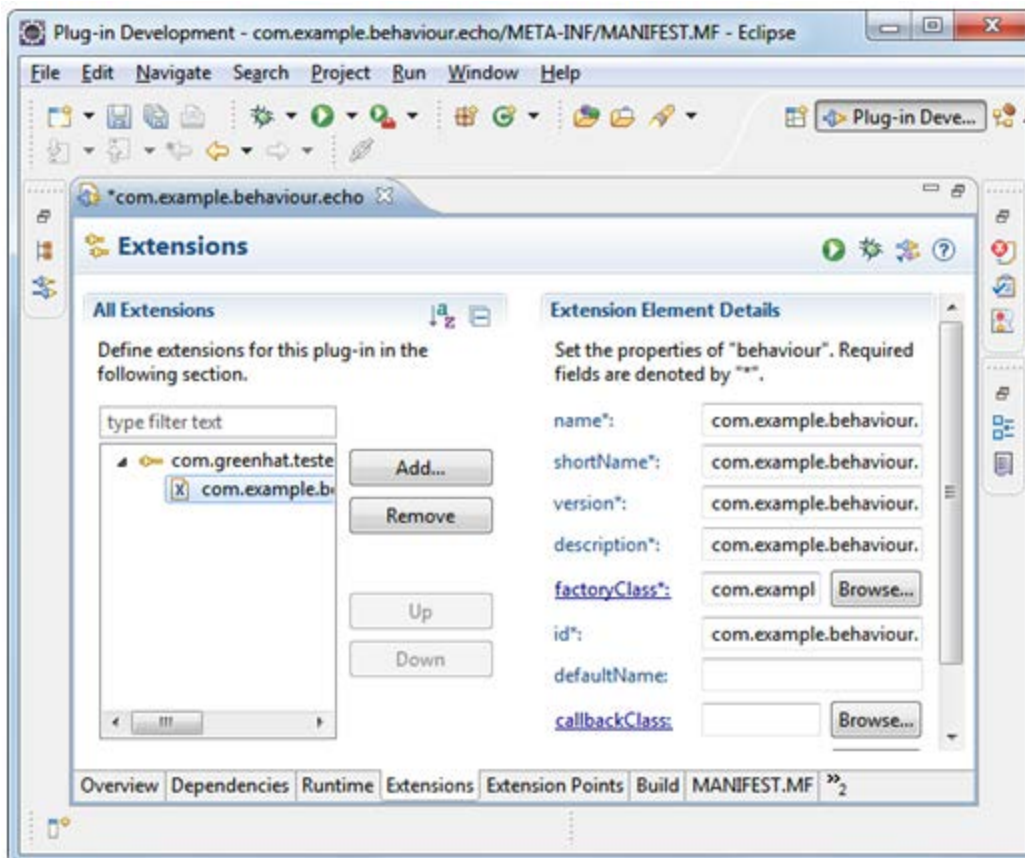
The Extension Point Selection dialog box is displayed.





30. On the **Extension Points** tab, click **com.greenhat.tester.api.behaviour**.
31. Click **Finish**.

The new extension point is displayed under **All Extensions**.

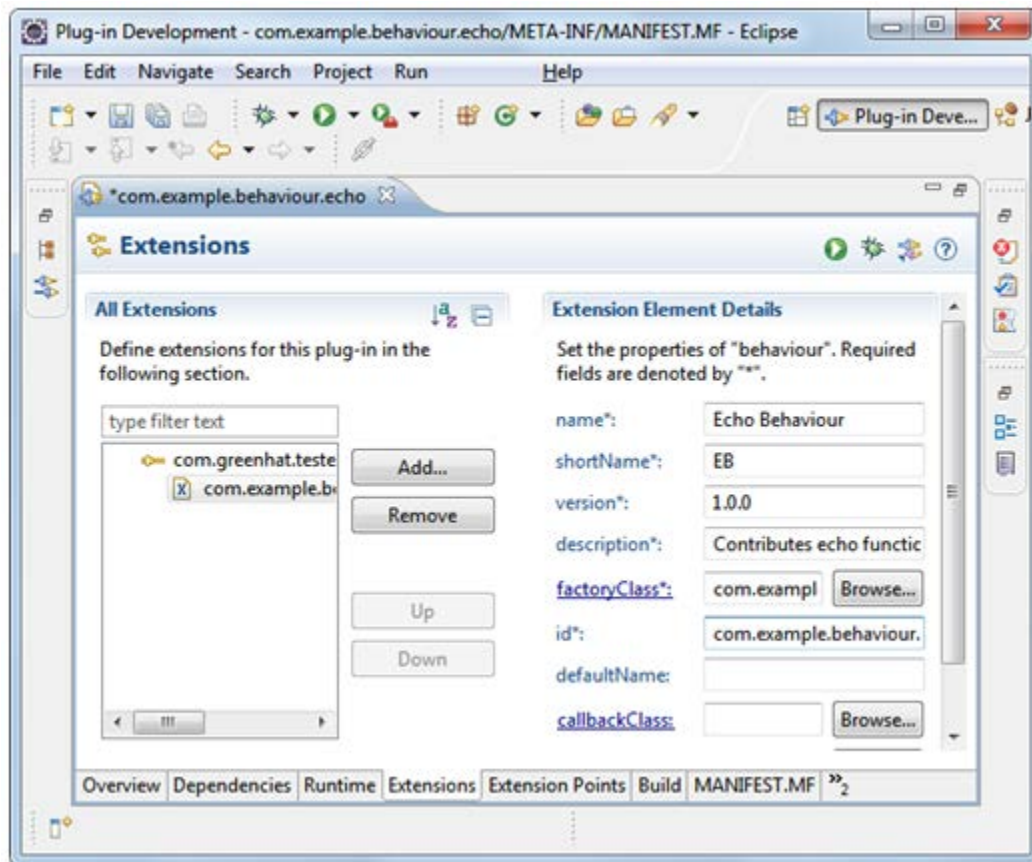


- 
32. Under **Extension Element Details**, enter the details provided in the following table.

---

Field	Description	Suggested Value
Name	The name of the behaviour.	Echo Behaviour
Short Name	A short name that helps to identify the behaviour.	EB
Version	This is used to determine the compatibility of persisted configuration.	1.0.0
Description	Description of the purpose and capabilities of the behaviour.	Contributes echo functionality
Factory Class	This identifies the class within the selected plugin that is capable of creating instances of the behaviour at run time if BehaviourFactory is implemented.	com.example.behaviour. .echo.EchoFactory
ID	Unique identifier for the behaviour.	com.example.behaviour. .echo

---



33. Click **File > Save**.
34. Define suitable interfaces for the behaviour and its callback interface. (For information about this, refer to [Defining Interfaces](#).)
35. Under **Extension Element Details** on the **Extensions** tab of the Manifest Editor window, enter the name of the callback interface in the **callbackClass** field.

For example: `com.example.behaviour.echo.EchoListener`.

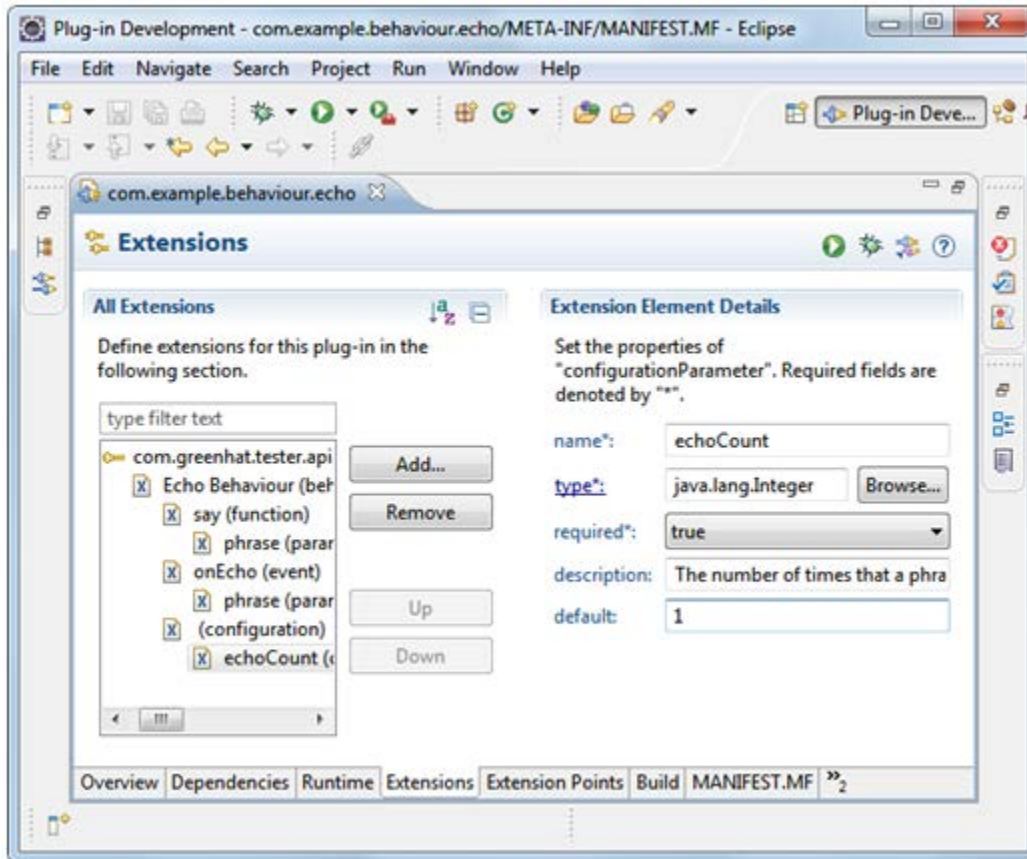
Alternatively, click the field's **Browse** button to select the class.

36. In the **behaviourClass** field, enter the name of the behaviour interface.

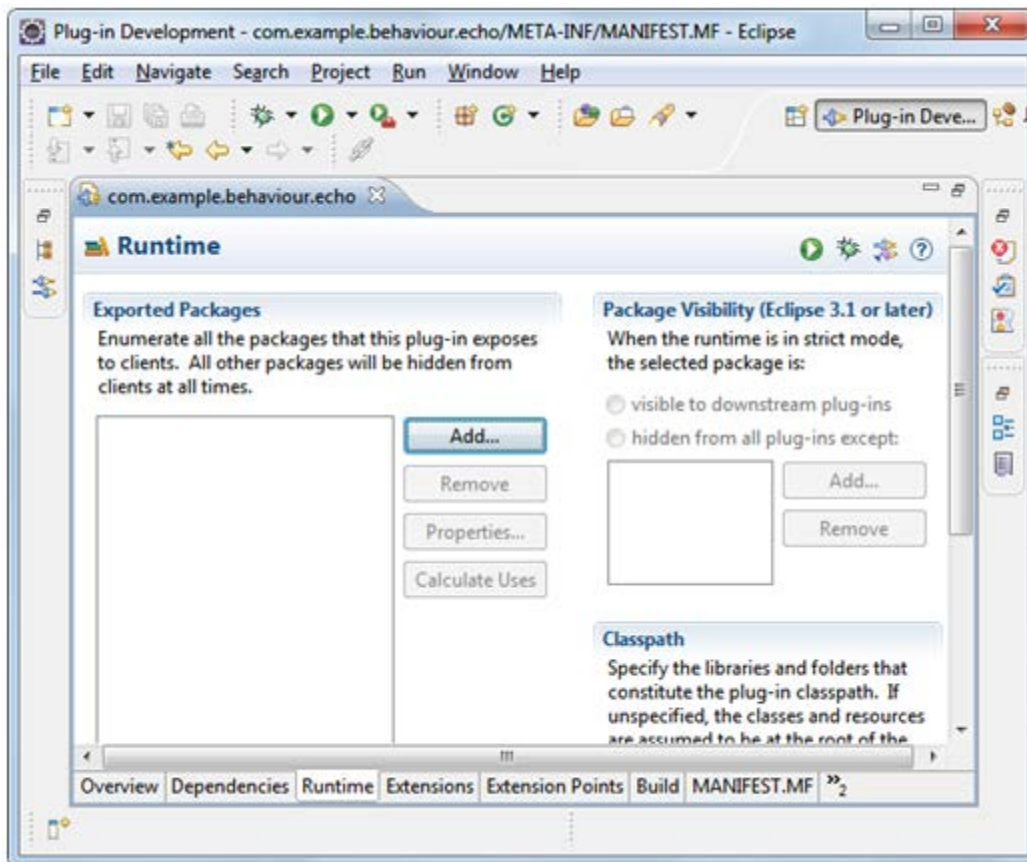
For example: `com.example.behaviour.echo.Echo`.

Alternatively, click the field's **Browse** button to select the class.

- 
37. Declare the exposed function and event with their parameters, and the single configuration parameter, within the echo behaviour element.

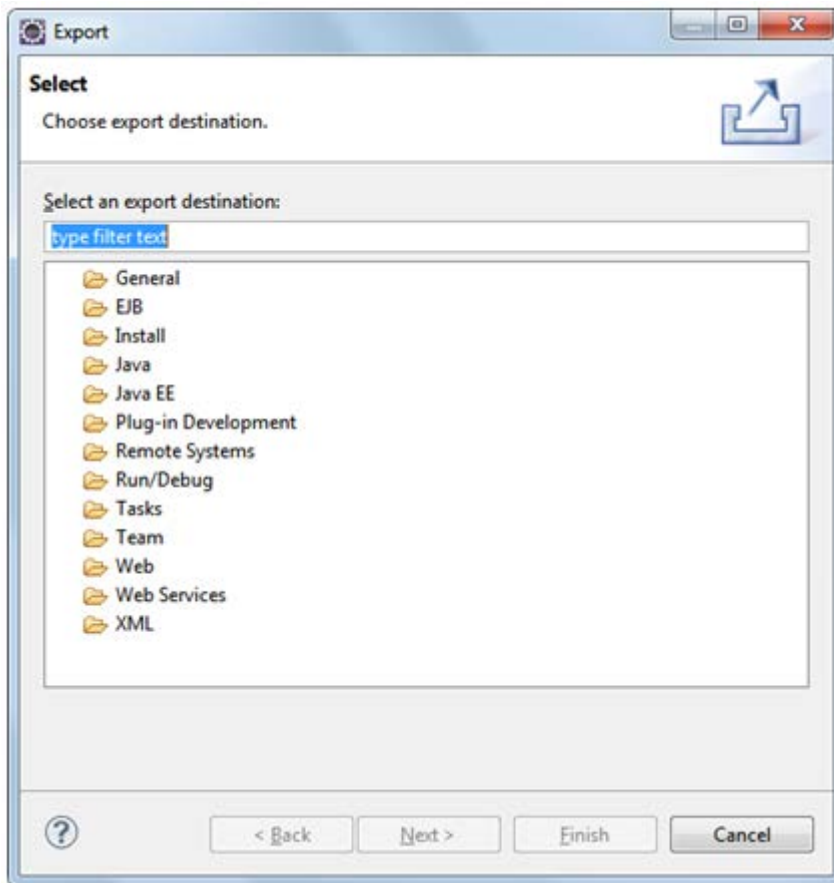


38. Click the **Runtime** tab.
- The **Runtime** tab is displayed.

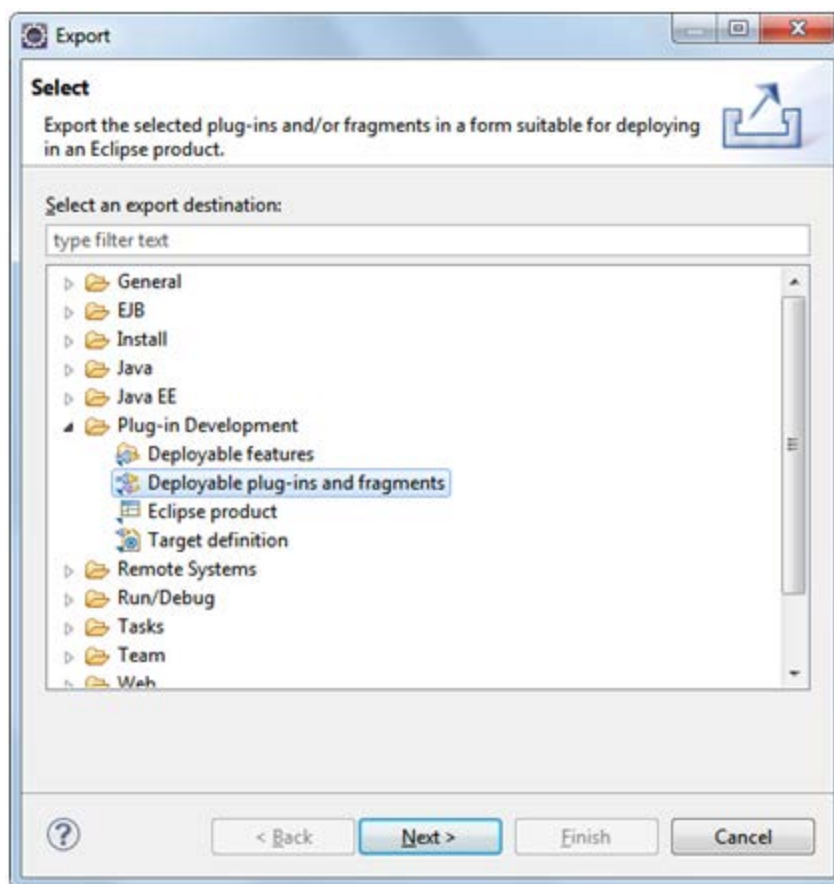


39. Ensure that the package containing the `EchoFactory` is exported from the bundle at run time.
40. Click **File > Save**.
41. Click **File > Export**.

The Export wizard is displayed.



42. Click **Plug-in Development > Deployable plug-ins and fragments**.



43. Click **Next**.

The second screen of the Export wizard is displayed.

44. Under **Available Plug-ins and Fragments**, select the plug-in's check box.

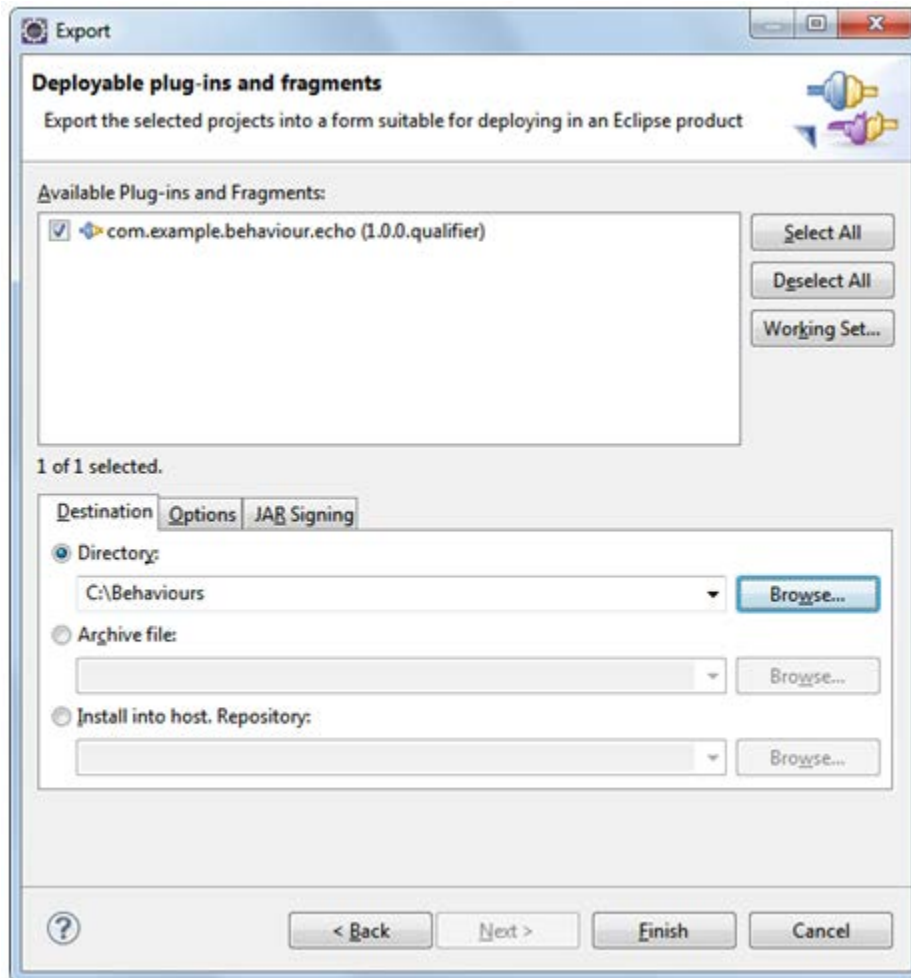
45. On the **Destination** tab, click the **Directory** option button.

46. In the **Directory** field, enter a directory path.



---

Alternatively, click the **Browse** button next to the **Directory** field and select the directory path.



47. Click **Finish**.

The plug-in is exported as a JAR file to a `plugins` folder in the specified directory. The name of the JAR file includes the current date and time.

48. Copy the JAR file to `<Rational Integration Tester Installation Directory>\plugins`.

This ensures that the next time when Rational Integration Tester is started, the echo behaviour will be available within the Stub Editor.

---

## 9.3 Defining Interfaces

The following sections show to define interfaces for the echo behaviour.

### 9.3.1 Behaviour Interface

```
package com.example.behaviour.echo;

import
com.greenhat.tester.api.behaviour.LifecycleAwareBehaviour;

public interface Echo extends LifecycleAwareBehaviour
{
    void say( String phrase );
}
```

### 9.3.2 Callback Interface

```
package com.example.behaviour.echo;

public interface EchoListener
{
    void onEcho( string phrase );
}
```

Rational Test Virtualization Server stubs will be able to invoke the `say(String phrase)` operation of this behaviour and can opt to handle the `onEcho(String phrase)` event.

Create a behaviour factory class and add an implementation for the new behaviour. The factory will retrieve configuration from the supplied configuration map and use it to parameterize the echo behaviour instance:

---

### 9.3.3 Behaviour Factory

```
package com.example.behaviour.echo;
import java.util.Map;
public class EchoFactory implements BehaviourFactory<Echo>
{
    public EchoFactory()
    {
    }
    @Override
    public Echo create( BehaviourServices services,
        Map<String, Object> configuration, Object callback )
    {
        String echoCountKey = "echoCountKey";
        int echoCount = 1;
        if ( configuration.containsKey( echoCountKey ) )
        {
            echoCount = (Integer)configuration.get(
                echoCountKey );
        }
        return new EchoImpl( (EchoListener)callback,
            echoCount );
    }
}
```

The echo behaviour implementation is shown in the following section.

---

### 9.3.4 Behaviour Implementation

```
package com.example.behaviour.echo.impl;
import java.util.concurrent.Executor;
public class EchoImpl implements Echo
{
    private final EchoListener callback;
    private final int echoCount;
    private final Executor executor =
        Executors.newSingleThreadExecutor();
    public EchoImpl( EchoListener callback, int
        echoCount )
    {
        this.callback = callback;
        this.echoCount = echoCount;
    }
    @Override
    public void say( final String phrase )
    {
        executor.execute( new Runnable()
        {
            @Override
            public void run()
            {
                for( int i = 0; i < echoCount; i++ )
                {
                    callback.onEcho( phrase );
                }
            }
        }));
    }
    //..
}
```

---

# Glossary

The following table below lists some of the key terms used in this document, and provides a description of each.

Term	Description
Agent	Agents run stubs in the Rational Test Virtualization Server environment and are deployed on one or more computers within the environment.
Field	A bit of data constituent to a message. Most fields are scalar and therefore unitary, equivalent to data attributes. Vector fields are an aggregation of fields both scalar and vector, and are usually referred to as Messages.
Message	A unit of information made up of a header consisting of meta-information and a body consisting of the message data.
Proxy	A proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers.
Server	A host computer on a network shared by more than one user.
Transport	Informally, the messaging software in use. For instance, TIBCO Rendezvous, TIBCO Active Enterprise, IBM WebSphere MQ, and HTTP.

---

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT,

---

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited  
Intellectual Property Law  
Hursley Park  
Winchester  
SO21 2JN  
Hampshire  
United Kingdom

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the

---

capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corporation 2001, 2012.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.



---

## Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

